Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Engenharia Elétrica

Compressão de Sequências Individuais e Fontes Parcialmente Ordenadas

Andresso da Silva

Área de Concentração: Processamento da Informação

Orientador: Prof. Dr. Francisco Marcos de Assis

Campina Grande, Paraíba, Brasil

Compressão de Sequências Individuais e Fontes Parcialmente Ordenadas

Andresso da Silva

Dissertação de Mestrado apresentada à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, em cumprimento às exigências para obtenção do Grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Engenharia Elétrica

Orientador: Prof. Dr. Francisco Marcos de Assis

Campina Grande, Paraíba, Brasil

S586c Silva, Andresso da.
Compressão de sequências individuais e fontes parcialmente ordenadas / Andresso da Silva. – Campina Grande, 2023.
100 f. : il. color.

Dissertação (Mestrado em Engenharia de Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia de Elétrica e Informática, 2022.
"Orientação: Prof. Dr. Francisco Marcos de Assis".
Referências.

1. Teoria da Informação. 2. Ordem Parcial. 3. Sistemas Concorrentes. 4. Monoides de Comutatividade. 5. Entropia. 6. Processamento da Informação. I. Assis, Francisco Marcos de. II. Título.

CDU 621.391(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA SEVERINA SUELI DA SILVA OLIVEIRA CRB-15/225

Compressão de Sequências Individuais e Fontes Parcialmente Ordenadas

Andresso da Silva

Dissertação Aprovada em 13 de Dezembro de 2022.

Francisco Marcos de Assis
Orientador

Benemar Alencar de Souza
Examinador

Eanes Torres Pereira
Examinador

Bruno Barbosa Albert
Examinador

Campina Grande - PB

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM ENGENHARIA ELETRICA
Rua Aprigio Veloso, 882, - Bairro Universitario, Campina Grande/PB, CEP 58429-900

REGISTRO DE PRESENÇA E ASSINATURAS

ATA DA DEFESA PARA CONCESSÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA, REALIZADA EM 13 DE DEZEMBRO DE 2022 **(N°749)**

CANDIDATO: **ANDRESSO DA SILVA**. COMISSÃO EXAMINADORA: BENEMAR ALENCAR DE SOUZA, D.Sc., UFCG, Presidente da Comissão e Examinador interno, FRANCISCO MARCOS DE ASSIS, Dr., UFCG, Orientador, EANES TORRES PEREIRA, Dr., UFCG, Examinador interno. TÍTULO DA DISSERTAÇÃO: Compressão de Sequências Individuais e Fontes Parcialmente Ordenadas. HORA DE INÍCIO: **14h00** – LOCAL: **Sala Virtual, conforme Art. 5º da PORTARIA SEI Nº 01/PRPG/UFCG/GPR, DE 09 DE MAIO DE 2022**. Em sessão pública, após exposição de cerca de 45 minutos, o candidato foi arguido oralmente pelos membros da Comissão Examinadora, tendo demonstrado suficiência de conhecimento e capacidade de sistematização, no tema de sua dissertação, obtendo conceito APROVADO. Face à aprovação, declara o presidente da Comissão, achar-se o examinando, legalmente habilitado a receber o Grau de Mestre em Engenharia Elétrica, cabendo à Universidade Federal de Campina Grande, como de direito, providenciar a expedição do Diploma, a que o mesmo faz jus. Na forma regulamentar, foi lavrada a presente ata, que é assinada por mim, Filipe Emmanuel Porfírio Correia, e os membros da Comissão Examinadora presentes. Campina Grande, 13 de Dezembro de 2022.

Filipe Emmanuel Porfírio Correia - Secretário(a)

BENEMAR ALENCAR DE SOUZA, D.Sc. , UFCG
Presidente da Banca e Examinador interno.

FRANCISCO MARCOS DE ASSIS, Dr., UFCG
Orientador

EANES TORRES PEREIRA, Dr., UFCG, Examinador interno

ANDRESSO DA SILVA
Candidato

2 - APROVAÇÃO

2.1. Segue a presente Ata de Defesa de Dissertação de Mestrado do candidato ANDRESSO DA SILVA, assinada eletronicamente pela Comissão Examinadora acima identificada.

2.2. No caso de examinadores externos que não possuam credenciamento de usuário externo ativo no SEI, para igual assinatura eletrônica, os examinadores internos signatários **certificam** que os examinadores externos acima identificados participaram da defesa da tese e tomaram conhecimento do teor deste documento.

Documento assinado eletronicamente por **FILIPE EMMANUEL PORFIRIO CORREIA**, **SECRETÁRIO (A)**, em 14/12/2022, às 10:54, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **FRANCISCO MARCOS DE ASSIS**, **PROFESSOR 3 GRAU**, em 14/12/2022, às 11:11, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **EANES TORRES PEREIRA**, **PROFESSOR 3 GRAU**, em 14/12/2022, às 11:28, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **2986565** e o código CRC **A15D180D**.

Referência: Processo nº 23096.087584/2022-17 SEI nº 2986565

Aos meus pais.

Agradecimentos

Aos meus pais por sempre terem dado todo o apoio e por serem minhas primeiras referências. Eles me ensinaram tudo o que eu poderia precisar e continuam ensinando.

A todos os professores que contribuíram para a minha formação, em especial ao professor Francisco por me apresentar o tema, me orientar e ser uma referência nos estudos.

A todos que contribuíram para o desenvolvimento desta dissertação, em especial Jayna, que sempre está disposta a revisar meus textos, mesmo soando como grego para ela.

Aos amigos do Instituto de Estudos em Computação e Informação Quânticas (IQuanta) por me receberem e pelas conversas, especialmente Milena, Micael e Thiciany, por me darem suporte quando precisei.

Ao IQuanta, pela estrutura física necessária para a realização desse trabalho. Ao Programa de Pós-Graduação em Engenharia Elétrica (PPgEE - COPELE) da UFCG, pelo suporte administrativo durante o período do mestrado. À CAPES, pelo suporte financeiro para o desenvolvimento dessa dissertação.

"Meu corpo nada quer, mas a minh'alma
em fogos de amplidão deseja tudo
o que ultrapassa o humano entendimento.
E embora nada atinja, não se acalma
e, sendo alma, transpõe meu corpo mudo,
e aos céus pede o inefável e não o vento."
Lêdo Ivo

Resumo

A maioria das aplicações da teoria da informação é baseada no uso de alfabetos com ordem total. Quando os sistemas considerados apresentam eventos com ordem parcial (*e.g.* sistemas concorrentes), a teoria da informação convencional fornece ferramentas incompletas devido à natureza dos processos. Isso se torna um assunto relevante ao considerar que o número de sistemas concorrentes tem crescido cada vez mais. Quando há ordem parcial, as principais ferramentas para a análise são os grafos de comutatividade e as classes de equivalência, induzidas por esses grafos. Nesta dissertação, busca-se desenvolver novas medidas de informação para alfabetos com ordem parcial e aplicar essas medidas na análise e desenvolvimento de algoritmos que utilizam esses alfabetos. Como resultados, foram obtidos métodos de estimação e limitantes para o número de classes de equivalência. Além disso, foi proposta uma nova definição de entropia para alfabetos com ordem parcial, chamada de entropia de comutatividade. É demonstrado que essa definição permite superar algumas das limitações de entropias semelhantes propostas anteriormente. Além disso, é apresentado um novo algoritmo de compressão ótimo que considera a ordem parcial entre símbolos da sequência a ser comprimida. Quando as relações de ordem parcial são consideradas, é demonstrado que é possível obter maiores taxas de compressão de sequências.

Palavras-chave: Ordem Parcial. Monoides de Comutatividade. Entropia. Algoritmos. Sistemas Concorrentes.

Abstract

Most applications of information theory are based on the use of alphabets with a total order. When the systems considered present events with partial order (e.g. concurrent systems), conventional information theory provides incomplete tools due to the nature of the processes. This becomes a relevant issue when considering that the number of concurrent systems has been growing more and more. When there is a partial order, the main tools for analysis are commutativity graphs and equivalence classes, induced by these graphs. It is aimed, in this dissertation, to develop new information measures for alphabets with partial order and apply these measures in the analysis and development of algorithms that use these alphabets. As a result, estimation methods and limits for the number of equivalence classes were obtained. Besides, it was proposed a new definition of entropy for alphabets with a partial order, called commutativity entropy. It is shown that this definition allows overcoming some of the limitations of similar entropies proposed before. Furthermore, a new optimal compression algorithm is presented that considers the partial order between symbols in the sequence to be compressed. When partial order relations are considered, it is shown that higher sequence compression rates can be obtained.

Keywords: Partial Order. Commutation Monoids. Entropy. Algorithms. Concurrent Systems.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

GPU	*Unidade gráfica de processamento*
AEP	*Propriedade de equipartição assintótica*
LZ77	*Algoritmo de Lempel-Ziv de 1977*
LZ78	*Algoritmo de Lempel-Ziv de 1978*
DST	*Árvore de busca digital*
POLZ	*Algoritmo de Lempel-Ziv com ordem parcial*
iid	*Independente e identicamente distribuído*
va	*Variável aleatória*
RTS	*Sistemas em tempo real*
CE	*Caixa eletrônico*
POSET	*Conjunto parcialmente ordenado*
VLSI	*Integração em grande escala*

Lista de Símbolos

X	Variável aleatória
$\mathbf{X}$	Processo estocástico
$H(\cdot)$	Entropia de Shannon
$H(\Sigma), H(\mathcal{X})$	Taxa de entropia
log	Log base 2
Σ, $\mathcal{X}$	Alfabeto finito
Σ^*	Conjunto de todas as palavras finitas definidas com Σ (monoide livre)
Σ^n	Conjunto de todas as palavras de comprimento n definidas em Σ
ε	Palavra vazia
$\mathbf{u}$,$\mathbf{v}$,$\mathbf{w}$	Palavras definida em um alfabeto
$\lvert\mathbf{u}\rvert$	Comprimento da palavra $\mathbf{u}$
σ	Um elemento de Σ
G	Grafo de comutatividade
$V(G), V$	Conjunto de vértices de um grafo
$E(G), E$	Conjunto de arestas de um grafo
$\overline{G}$	Grafo de não-comutatividade
$\equiv_G$	Relação de congruência definida por um grafo de comutatividade
$\mathcal{E}_G(\mathbf{u})$	Conjunto de palavras equivalentes a $\mathbf{u}$ de acordo com $\equiv_G$
P_u	Tipo da palavra $\mathbf{u}$
$\pi_A(\mathbf{u})$	Projeção de $\mathbf{u}$ no conjunto A
$N(\sigma\lvert\mathbf{u})$	Número de ocorrência de σ em $\mathbf{u}$

$\tau_G(n)$	Número de classes de equivalência de comprimento n de acordo com G
$D(G, z)$	Polinômio de dependência do grafo G
$\beta(G)$	Inverso da raiz de menor módulo de $D(G, z)$
$\mathcal{M}(\Sigma, G)$	Monoide de comutatividade gerado a partir do monoide livre Σ^* e a relação $\equiv_G$
$\zeta_{\mathcal{M}}(z)$	Função geradora do monoide de comutatividade
$K(\mathbf{u})$	Complexidade Kolmogorov de $\mathbf{u}$
$C_\iota(G, \mathbf{u})$	Complexidade de intercâmbio de $\mathbf{u}$ em relação ao grafo G
$H_\iota(G, P)$	Entropia de intercâmbio do grafo G com distribuição de probabilidade P nos vértices
$H(G, P)$	Entropia de comutatividade do grafo G com distribuição de probabilidade P nos vértices
$S(G, P)$	Fonte parcialmente comutativa definida no grafo de comutatividade G e com distribuição P.
$C(S)$	Complexidade de uma sequência S, de acordo com o algoritmo LZ78
$C(G, S)$	Complexidade de uma sequência S, de acordo com o algoritmo POLZ
$O(g(x))$	Notação grande-O
$\wedge$	Conjunção, "e" lógico
$(\mathcal{P}, \leq_P)$	Conjunto parcialmente ordenado definido no conjunto $\mathcal{P}$ e relação de ordem $\leq_P$.

Sumário

Capítulo 1

Introdução

Neste capítulo, serão apresentadas a motivação e a relevância da pesquisa desenvolvida nesta dissertação. São apresentadas a notação e a terminologia, visando padronizar as discussões subsequentes, apresentando possíveis simplificações na notação que é complementada pela lista de símbolos. As contribuições também são apresentadas, sintetizando o que foi desenvolvido nesta dissertação. Por fim, é apresentada a organização do texto desta dissertação.

1.1 Introdução

Desde a publicação do trabalho seminal de Shannon [1], a teoria da informação vem sendo aplicada com sucesso em diversos cenários. As aplicações vão de campos como teoria da comunicação até biologia. Apesar de campos distintos, a maioria das aplicações é baseada em modelagem usando sequências definidas em alfabetos com ordem total.

O mesmo pode ser observado em algoritmos que utilizam resultados da teoria da informação, como algoritmos de compressão, de codificação e de processamento de dados. Quando as aplicações não podem ser completamente modeladas por meio da abordagem sequencial, a teoria da informação convencional ainda pode ser utilizada, mas não traz informações intrínsecas ao comportamento da ordem parcial.

Um dos exemplos de sistemas que não são completamente modelados com uma abordagem sequencial são os sistemas concorrentes. Esses sistemas são compostos por um subconjunto de processos que podem funcionar em paralelo uns com os outros sem alterar o resultado final; enquanto alguns dos processos precisam ser executados em uma ordem determinada.

Mazurkiewicz [2] utilizou o conceito de ordem parcial para denotar ocorrências de processos concorrentes como palavras, em que cada símbolo representa um processo. Com estas palavras, dois símbolos contíguos podem trocar de posição se representarem dois processos que podem ser executados em paralelo. A troca de posição desses símbolos define

uma relação de comutatividade parcial que gera palavras equivalentes. O conjunto destas palavras é conhecido como traços de Mazurkiewicz ou, doravante, classes de equivalência.

As classes de equivalência podem ser a ferramenta fundamental para definir medidas de informação para sequências definidas em alfabetos com ordem parcial. Generalizar as medidas de informação para esses alfabetos possibilitará analisar processos gerais de forma mais acurada. Além disso, também possibilitará desenvolvimento e avaliação de novos algoritmos, possivelmente mais eficientes em algum aspecto. Desta forma, no presente trabalho, busca-se definir novas medidas de informação considerando alfabetos com ordem parcial e desenvolver novos algoritmos de compressão voltados para esses alfabetos.

1.2 Justificativa e Relevância

A utilização de sistemas concorrentes tem crescido cada vez mais, tendo como exemplos as redes de computadores, computação distribuída, sistemas operacionais, computação em nuvem, unidades gráficas de processamento (GPU) e diversos outros [3, 4]. Entretanto, a maioria dos algoritmos e técnicas de análise ainda é voltada para aplicações sequenciais.

Sistemas concorrentes podem ser naturalmente modelados por meio da utilização de ordem parcial. Portanto, o desenvolvimento deste trabalho tem como objetivo estudar os métodos relacionados aos alfabetos com ordem parcial e utilizar medidas da teoria da informação para analisar e desenvolver técnicas voltadas à concorrência.

O desenvolvimento de técnicas que consideram ordem parcial é de fundamental importância, pois possibilitam as análises que as técnicas sequenciais falham ou são limitadas. Além disso, o desenvolvimento dessas técnicas pode fornecer pistas de como criar algoritmos mais eficientes utilizando a concorrência.

1.3 Notação e Terminologia

Variáveis aleatórias serão representadas por meio de letras maiúsculas, processos estocásticos com letras maiúsculas em negrito, seus alfabetos com letras caligráficas (por exemplo, $\mathcal{X}$ denota o alfabeto da variável aleatória X) e sua função de densidade de probabilidade com $p(\cdot)$. Alfabetos também serão representados por Σ, quando se tratar de alfabetos com ordem total ou parcial e será feita a distinção, quando necessário. Ao longo de toda a dissertação, log refere-se à base 2 e a informação medida será em *bits*. Seja $\mathcal{V}$ um conjunto, $\mathcal{V} \times \mathcal{V}$ representa o produto cartesiano de $\mathcal{V}$ com ele mesmo. Nesta dissertação, sequências e palavras serão usadas com o mesmo significado, assim como sub-sequências e sub-palavras. Além disso, quando se falar somente em monoide, se refere ao monoide de comutatividade, a não ser que seja explicitamente especificado como outro monoide.

Teoremas apresentados na fundamentação e na revisão bibliográfica serão enunciados sem provas.

1.4 Contribuições

As principais contribuições nesta dissertação que a distingue de outros trabalhos estão listadas abaixo:

1. Obtenção de uma aproximação assintótica do número de classes de equivalência a partir do grafo de comutatividade.
2. Obtenção de limitantes para o número de classes de equivalência para famílias de grafos de comutatividade com $|V|$ vértices e $|E|$ arestas.
3. Aplicação de teoria dos conjuntos parcialmente ordenados para obter limitantes no número de elementos de uma classe de equivalência.
4. Apresentação de uma nova definição de entropia que considera alfabetos com ordem parcial, chamada de entropia de comutatividade e que supera limitações das propostas anteriores.
5. Apresentação de um algoritmo ótimo para a compressão de sequências definidas em alfabetos com ordem parcial.

1.5 Organização do texto

Esta dissertação está estruturada da maneira como segue. No capítulo 2, é apresentado o embasamento teórico necessário para o desenvolvimento dos estudos envolvendo alfabetos com ordem parcial e para o entendimento da dissertação. Ainda nesse capítulo, são desenvolvidos novos métodos para a estimação do número de classes de equivalência e limitantes relacionados. No capítulo 3, é apresentado uma revisão bibliográfica acerca das entropias aplicadas em alfabetos com ordem parcial e é apresentada uma nova definição de entropia, chamada de entropia de comutatividade. No capítulo 4, é apresentado uma revisão bibliográfica envolvendo algoritmos de compressão de sequências, dando foco àqueles voltados para alfabetos com ordem parcial. Também nesse capítulo, é apresentado um novo algoritmo para a compressão de sequências definidas nesses alfabetos e, utilizando os resultados desenvolvidos no Capítulo 2, é demonstrado que ele é um algoritmo ótimo. Por fim, são apresentadas as conclusões no capítulo 5. Após o Capítulo 5, estão as referências seguidas dos apêndices que contemplam temas como teoria dos grafos, conjuntos parcialmente ordenados, teoria da informação e teoria da complexidade.

Capítulo 2

Limitantes para o Número de Classes de Equivalência

Neste capítulo serão apresentados os fundamentos da teoria dos monoides de comutatividade e da teoria dos traços. Também serão apresentados os fundamentos da teoria de pilha de peças que permite uma interpretação geométrica dos monoides de comutatividade. Esses temas serão utilizados para determinar um novo método de estimação do número de classes de equivalência dos monoides de comutatividade e para estimar o número de elementos pertencentes a uma mesma classe de equivalência. Este capítulo utiliza frequentemente temas abordados nos Apêndices A e B.

2.1 Introdução

Cartier e Foata [5] desenvolveram propriedades combinatórias de sequências definidas em alfabetos finitos com comutatividade parcial. Desta forma, dois elementos a e b desse alfabeto podem ter a propriedade de que $ab = ba$. Quando dois elementos possuem essa relação, é dito que eles comutam. Essa comutatividade permite que algumas sequências de símbolos sejam equivalentes a outras, formando as classes de equivalência. Os monoides com essas características são agora conhecidos como monoides de Cartier-Foata ou monoides de comutatividade.

Mazurkiewicz [2] utilizou os monoides de comutatividade para desenvolver a teoria dos traços. Essa abordagem introduziu a representação de eventos concorrentes como uma cadeia sequencial de símbolos. Utilizando a linguagem dos monoides de comutatividade, se dois eventos são concorrentes, então os símbolos associados a esses eventos comutam. A teoria dos traços foi desenvolvida com o intuito de fornecer ferramentas de linguagens formais à análise de sistemas concorrentes e, em particular, de redes de Petri. Com essa representação, é possível analisar o comportamento de sistemas concorrentes e sequenciais. O que Mazurkiewic chamou de traços, são as classes de equivalência na terminologia

abordada na presente dissertação.

Viennot [6] desenvolveu o conceito de pilha de peças que é equivalente aos monoides de comutatividade, mas com a possibilidade de interpretação geométrica e visualização. Essa interpretação, permitiu resolver problemas de física estatística e estimar o número de palavras que pertencem a determinada classe de equivalência.

Fisher [7] e, depois, Fisher e Solow [8] desenvolveram, a partir do trabalho de Cartier e Foata, um modo de determinar o número exato de classes de equivalência a partir do grafo de comutatividade. Conhecer o número de classes de equivalência se mostrou útil nesta dissertação para a análise dos algoritmos desenvolvidos e que serão apresentados nos próximos capítulos.

Nas próximas seções desse capítulo, serão abordados os monoides de comutatividade e as pilhas de peças. Esses são os conceitos fundamentais para desenvolver e entender os resultados obtidos nesta dissertação.

2.2 Monoide de Comutatividade

Seja Σ um alfabeto finito e que Σ^* seja o conjunto de todas as palavras finitas formadas pelos elementos de Σ, incluindo a palavra vazia ε. O conjunto Σ^* é chamado de monoide livre gerado por Σ. Além disso, seja Σ^n o conjunto de todas as palavras de comprimento n definidas no alfabeto Σ.

Seja $G = (V, E)$ um grafo simples não-orientado chamado *grafo de comutatividade*, no qual V é o conjunto de vértices e E é o conjunto de arestas (ver Apêndice A para mais detalhes). Os elementos de V estão associados a um alfabeto finito Σ por meio de uma função bijetiva $\lambda : V \mapsto \Sigma$.

Dois símbolos de Σ comutam quando há duas palavras $\mathbf{u} = \mathbf{l}xy\mathbf{m}$ e $\mathbf{v} = \mathbf{l}yx\mathbf{m}$ que representam execução equivalente de sistemas concorrentes, com $\mathbf{l}, \mathbf{m} \in \Sigma^*$ e $x, y \in \Sigma$. Esses tipos de palavras são conhecidos como traços de Mazurkiewicz [2].

Se os símbolos $\lambda(x), \lambda(y) \in \Sigma$ comutam, então os vértices $x, y \in V$ estão conectados por uma aresta, $(x, y) \in E$. Se dois vértices estão conectados por uma aresta, diz-se que eles são adjacentes. O complemento do grafo de comutatividade, $\overline{G} = (V, \overline{E})$, é chamado de grafo de não-comutatividade, no qual os vértices conectados por uma aresta estão associados aos símbolos que *não* comutam.

Se os símbolos $\lambda(x)$ e $\lambda(y)$ comutam, então a relação é representada por $xy \equiv_G yx$. Se eles não comutam, a relação é representada por $xy \not\equiv_G yx$. Outra maneira de representar isso é por meio de $xy \equiv_G yx \Leftrightarrow (x, y) \in E$. Por exemplo, se o *grafo de comutatividade* for $a - b - c$, então a comuta com b e b comuta com c, mas a não comuta com c. Portanto, nesse caso, $ab \equiv_G ba$, $bc \equiv_G cb$ e $ac \not\equiv_G ca$. Desta forma, pode-se definir o monoide de

comutatividade.

Definição 2.1 (Monoide de Comutatividade)**.** *O monoide de comutatividade* $\mathcal{M}(\Sigma, G) = \Sigma^* \backslash \equiv_G$ *é o quociente do monoide livre* Σ^* *pela relação de congruência* $\equiv_G$*.*

Duas palavras $\mathbf{u}, \mathbf{v} \in \mathcal{M}(\Sigma, G)$ são equivalentes de acordo com a relação $\equiv_G$ se for possível obter uma a partir da outra trocando as posições dos símbolos consecutivos que comutam.

Definição 2.2 (Classe de equivalência)**.** *Seja* $\mathcal{E}_G(\mathbf{u})$ *o conjunto de palavras equivalentes a uma palavra* $\mathbf{u} \in \mathcal{M}(\Sigma, G)$ *de acordo com a relação* $\equiv_G$*. O conjunto* $\mathcal{E}_G(\mathbf{u})$ *é chamado de classe de equivalência de* $\mathbf{u}$*.*

Se duas palavras pertencem à mesma classe de equivalência, diz-se que elas são congruentes. Quando não houver possibilidade de ambiguidade, uma classe de acordo com um grafo de comutatividade será representada somente por $\mathcal{E}(\mathbf{u})$.

Para um determinado comprimento, existirá um número de classes de equivalência. O número de classes será denotado por $\tau_G(n)$, em que G é o grafo de comutatividade e n é o comprimento das palavras. Desta forma, $\tau_G(n)$ representa o número de palavras de comprimento n que *não são equivalentes* entre si, podendo ser entendidas como o número de mensagens distintas possíveis de serem enviadas por uma fonte.

Exemplo 2.1. *Neste exemplo, será considerado o grafo de comutatividade da Fig. 2.1.*

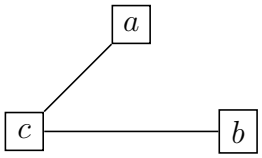


Figura 2.1 – Grafo de comutatividade G.

Desta forma, $ac \equiv_G ca$*,* $cb \equiv_G bc$ *e* $ab \not\equiv_G ba$*. As classes de comprimento* $n = 1$ *serão* $\{a, b, c\}$*, totalizando 3 classes de equivalência. Para um comprimento de* $n = 2$*, as classes serão* $\{aa, ab, \{ac, ca\}, ba, bb, \{bc, cb\}, cc\}$*, com um total de 7 classes. Agora para um comprimento de* $n = 3$ *as classes serão* $\{aaa, aab, \{aac, aca, caa\}, aba, abb, \{abc, acb, cab\}, \{acc, cac, cca\}, baa, bab, \{bac, bca, cba\}, bba, bbb, \{bbc, bcb, cbb\}, \{bcc, cbc, ccb\}, ccc\}$*, sendo composta por 15 classes.*

Cabe notar que se G fosse vazio (i.e., nenhum símbolo comuta), as classes de comprimento n teriam $|\Sigma|^n$ elementos. Sendo assim, é possível concluir que, de forma geral, $\tau_G(n) \leq |\Sigma|^n$, sendo a igualdade alcançada quando nenhum símbolo comuta. Isso significa que, dado a comutação de pelo menos dois símbolos, o número de palavras distintas cresce

mais lentamente no caso dos monoides de comutatividade, pois $\tau_G(n) < |\Sigma|^n$. O problema de contar o número de classes de equivalência será abordado na próxima seção.

A fim de verificar se duas palavras $\mathbf{u}$ e $\mathbf{v}$ são congruentes, uma abordagem possível seria gerar todas as palavras equivalentes a $\mathbf{u}$ e verificar se $\mathbf{v}$ está dentro desse conjunto. Entretanto, gerar as classes de equivalência das palavras pode ser custoso computacionalmente, pois é função do número de símbolos na palavra, seu comprimento e as relações de comutatividade definidas pelo grafo. Desta forma, pode-se verificar se duas palavras pertencem à mesma classe de equivalência a partir do teorema de Perrin [9]. Para compreender esse teorema, é necessário definir o tipo de uma palavra e a projeção de uma palavra nas arestas de um grafo.

Definição 2.3. *(Projeção nas arestas de um grafo [9]) Para qualquer subconjunto A do alfabeto Σ e qualquer palavra $\mathbf{w}$ definida em Σ^*, a projeção $\pi_A(\mathbf{w})$ da palavra w em A é obtida deletando de $\mathbf{w}$ todos os símbolos que não estão presentes em A.*

Duas palavras $\mathbf{u}$ e $\mathbf{v}$ têm o mesmo tipo se elas possuem a mesma frequência de ocorrência de símbolos. Desta forma, $N(\sigma|\mathbf{u}) = N(\sigma|\mathbf{v})$ para todos os $\sigma \in \Sigma$ (ver Apêndice C), sendo $N(\sigma|\mathbf{u})$ o número de ocorrências de σ em $\mathbf{u}$. Dito isto, uma condição necessária é que, para duas palavras terem o mesmo tipo, elas devem possuir o mesmo comprimento. O teorema de Perrin [9] é enunciado a seguir.

Teorema 2.1. *(Perrin [9, p.330]) As condições necessárias e suficientes para que duas palavras $\mathbf{w}, \mathbf{u}$ sejam congruentes é que elas possuam o mesmo tipo e que $\pi_{\{x,y\}}(\mathbf{w}) = \pi_{\{x,y\}}(\mathbf{u})$ para todos os símbolos $x, y \in \Sigma$ que são adjacentes no grafo de não-comutatividade $\overline{G}$.*

Exemplo 2.2. *Considere como exemplo o grafo de não-comutatividade $\overline{G}$ apresentado na Fig. 2.2. Desta forma, o alfabeto é $\Sigma = \{a, b, c, d, e\}$. As palavras $\mathbf{u} = abcde$, $\mathbf{v} = cabed$ e $\mathbf{w} = adbec$ estão definidas em Σ^5 e possuem o mesmo tipo $P_u = P_v = P_w = \left\{\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right\}$. A classe de equivalência de $\mathbf{u}$ é igual a $\mathcal{E}_G(abcde) = \{abcde, abced, acbed, acbde, cabde, cabed, acbed, acebd, caebd\}$. A palavra $\mathbf{u}$ é congruente a $\mathbf{v}$, mas não é congruente a $\mathbf{w}$. Isso pode ser concluído porque $cabed \in \mathcal{E}_G(abcde)$, mas $adbec \notin \mathcal{E}_G(abcde)$.*

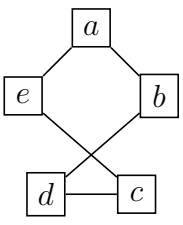


Figura 2.2 – Grafo de não-comutatividade $\overline{G}$.

Outra forma de verificar as relações de congruência entre as palavras u, v, w é utilizar o Teorema 2.1. O primeiro passo é verificar se as palavras possuem o mesmo tipo. Após isso, é necessário identificar os pares de símbolos adjacentes em $\overline{G}$ que são (a, b), (a, e),

(b,d), (c,d), (c,e). *Por conseguinte, as projeções de* $\mathbf{u}$ *nas duas primeiras arestas serão de* $\pi_{\{a,b\}}(\mathbf{u}) = ab$, $\pi_{\{a,e\}}(\mathbf{u}) = ae$. *Na Tabela 2.1 são apresentadas as projeções de* $\mathbf{u}$, $\mathbf{v}$ *e* $\mathbf{w}$ *em todas as arestas de* $\overline{G}$.

Aresta	$\pi_{\{x,y\}}(abcde)$	$\pi_{\{x,y\}}(cabed)$	$\pi_{\{x,y\}}(adbec)$
(a,b)	ab	ab	ab
(a,e)	ae	ae	ae
(b,d)	bd	bd	db
(c,d)	cd	cd	dc
(c,e)	ce	ce	ec

Tabela 2.1 – Projeção de $\mathbf{u}$, $\mathbf{v}$ e $\mathbf{w}$ nas arestas de $\overline{G}$.

Como em todos os casos as projeções de $\mathbf{u}$ *e* $\mathbf{v}$ *são iguais, então elas são congruentes. Entretanto,* $\mathbf{u}$ *e* $\mathbf{u}$ *não são congruentes a* $\mathbf{w}$.

2.3 Contagem de Classes de Equivalência

O número $\tau_G(n)$ de classes de equivalência de comprimento n considerando a relação $\equiv_G$ corresponde ao número de palavras $\mathbf{u}_i, i = 1,2,\ldots,\tau_G(n)$ definidas no monoide $\mathcal{M}(\Sigma, G)$ e de comprimento n que são não-congruentes par a par. Desta forma, representando o subconjunto de palavras do monoide $\mathcal{M}(\Sigma, G)$ que possuem comprimento n por $\mathcal{M}^n(\Sigma, G) = \Sigma^n \setminus \equiv_G$, então

$$\mathcal{M}^n(\Sigma, G) = \mathcal{E}_G(\mathbf{u}_1) \cup \mathcal{E}_G(\mathbf{u}_2) \cup \cdots \cup \mathcal{E}_G(\mathbf{u}_{\tau_G(n)}), \tag{2.1}$$

em que $\mathcal{E}_G(\mathbf{u}_i) \cap \mathcal{E}_G(\mathbf{u}_j) = \emptyset$ para $i \neq j$, $|\mathbf{u}_i| = n, i = 1,2,\ldots,\tau_G(n)$ e, por fim, $\tau_G(n) = |\mathcal{M}^n(\Sigma, G)|$.

Exemplo 2.3. *Considerado, como exemplo o grafo de comutatividade do exemplo 2.1, foi visto que* $ac \equiv_G ca$, $cb \equiv_G bc$ *e* $ab \not\equiv_G ba$. *Desta forma,*

- $n = 1$*:* $\mathcal{M}^1(\Sigma, G) = \{a, b, c\}$ *e* $\tau_G(n) = 3$*;*
- $n = 2$*:* $\mathcal{M}^2(\Sigma, G) = \{aa, ab, \{ac, ca\}, ba, bb, \{bc, cb\}, cc\}$ *e* $\tau_G(n) = 7$*;*
- $n = 3$*:* $\mathcal{M}^3(\Sigma, G) = \{aaa, aab, \{aac, aca, caa\}, aba, abb, \{abc, acb, cab\}, \{acc, cac, cca\}, baa, bab, \{bac, bca, cba\}, bba, bbb, \{bbc, bcb, cbb\}, \{bcc, cbc, ccb\}, ccc\}$ *e* $\tau_G(n) = 15$.

Fisher [7, 8] desenvolveu métodos para a determinação do número $\tau_G(n)$ de classes de equivalência de comprimento n com base na teoria de monoides parcialmente comutativos apresentada por Cartier e Foata [5]. As principais ferramentas para determinar $\tau_G(n)$ são o polinômio de dependência de um grafo de comutatividade G e a função geradora de classes de equivalência do monoide [8].

Definição 2.4. *(Polinômio de Dependência [8]) O polinômio de dependência do grafo de comutatividade G é definido por*

$$D(G, z) = \sum_{k=0}^{\omega} (-1)^k c_k z^k, \tag{2.2}$$

em que c_k denota o número de sub-grafos completos (cliques) de tamanho k no grafo G e ω é o número do clique de G (tamanho do maior clique). Cabe notar que os valores de c_k são também iguais ao número de anti-cliques de tamanho k do grafo de não-comutatividade $\overline{G}$.

A função geradora $\zeta_{\mathcal{M}}$ do monoide $\mathcal{M}(\Sigma, G)$ pode ser utilizada para a obtenção de uma expressão para $\tau_G(n)$ a partir do Corolário 2.1. É possível também utilizá-la para analisar o comportamento assintótico das sequências $\{\tau_G(n)\}_{n\geq 0}$.

Definição 2.5. *(Função Geradora de um Monoide) A função geradora de um monoide $\mathcal{M}(\Sigma, G)$ é definida como*

$$\zeta_{\mathcal{M}}(z) = \sum_{n=0}^{\infty} \tau_G(n) z^n, \tag{2.3}$$

em que $\tau_G(n)$ é o número de classes de equivalência de comprimento n.

Corolário 2.1. *[8, p.251] A função geradora $\zeta_{\mathcal{M}}(z)$ do monoide é igual ao inverso do polinômio de dependência do grafo de comutatividade, ou ainda,*

$$\zeta_{\mathcal{M}}(z) D(G, z) = 1. \tag{2.4}$$

Exemplo 2.4. *Para ilustrar alguns dos conceitos, será considerado o grafo de comutatividade G da Fig.2.3.*

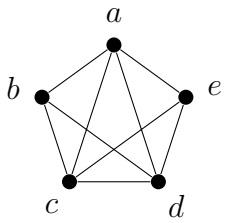
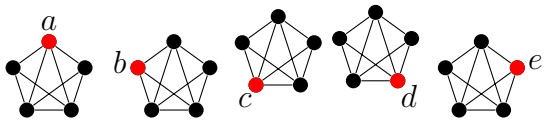
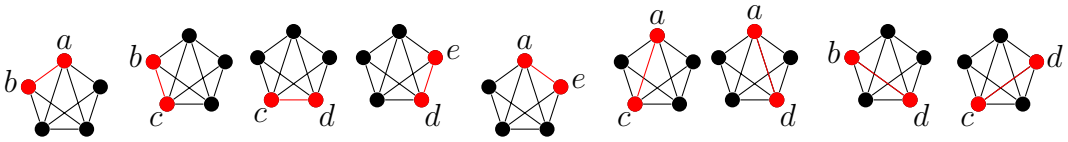


Figura 2.3 – Grafo de comutatividade G.

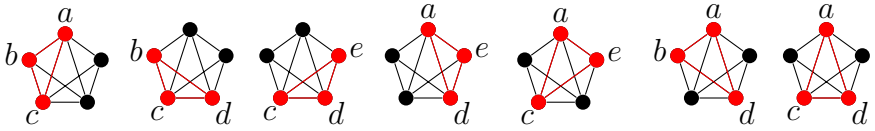
Na Fig.2.4 são apresentados os cliques de G. Como G não é completo, ele não possui subgrafos de tamanho 5, de forma a ter um número do clique igual a $\omega = 4$.



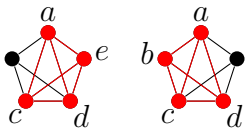
(a) Subgrafos completos de tamanho $k = 1$.



(b) Subgrafos completos de tamanho $k = 2$.



(c) Subgrafos completos de tamanho $k = 3$.



(d) Subgrafos completos de tamanho $k = 4$.

Figura 2.4 – Subgrafos completos do grafo de comutatividade G.

O conjunto de cliques pode ser representado por

$$K(G) = \{(\emptyset), (a, b, c, d, e), (ab, bc, cd, de, ac, ad, bd, cd, ce),$$
$$(abc, bcd, cde, dea, ace, abd, acd), (acde, abcd)\}$$

Desta forma, $c_0 = 1$, $c_1 = 5$, $c_2 = 9$, $c_3 = 7$, $c_4 = 2$ *e o polinômio de dependência de* G *é*

$$D(G, z) = \sum_{k=0}^{\omega} (-1)^k c_k z^k = 1 - 5z + 9z^2 - 7z^3 + 2z^4$$

E a função geradora do monoide será dada por

$$\zeta_{\mathcal{M}}(z) = \frac{1}{D(G, z)} = 1 + 5z + 16z^2 + 42z^3 + 99z^4 + 219z^5 + \ldots$$

Portanto, $\tau_G(0) = 1$, $\tau_G(1) = 5$, $\tau_G(2) = 16$, $\tau_G(3) = 42$, $\tau_G(4) = 99$ *e assim por diante. Um modo de encontrar uma expressão explícita para* $\tau_G(n)$ *de forma analítica é fazer a expansão de* $\zeta_{\mathcal{M}}(z)$ *em frações parciais e calcular a série de Taylor para cada um dos termos. Assim,* $\zeta_{\mathcal{M}}(z)$ *do exemplo pode ser decomposta em*

$$\begin{aligned}\zeta_{\mathcal{M}}(z) &= -\frac{8}{2z-1} + \frac{4}{z-1} - \frac{2}{(z-1)^2} + \frac{1}{(z-1)^3} \\ &= \sum_{n=0}^{\infty} 2^{n+3} z^n + \sum_{n=0}^{\infty} (-4) z^n + \sum_{n=0}^{\infty} (-2)(1+n) z^n + \sum_{n=0}^{\infty} \left(\frac{-1}{2}\right) (n+1)(n+2) z^n \\ &= \sum_{n=0}^{\infty} \left(2^{n+3} - 0{,}5n^2 - 3{,}5n - 7\right) z^n.\end{aligned}$$

Assim,

$$\tau_G(n) = 2^{n+3} - 0{,}5n^2 - 3{,}5n - 7.$$

Exemplo 2.5. *Como exemplo, será utilizado um grafo de não-comutatividade completo $\overline{G}$ e um alfabeto $\Sigma = \{a, b, c, d\}$. Um grafo completo é um grafo simples em que todo par de vértices é conectado por uma aresta. O polinômio de dependência, nesse caso, terá $c_1 = 4$ e $c_i = 0, i = 2, 3, 4$, ou seja, $D(G, z) = 1-4z$. Além disso, $\zeta_{\mathcal{M}}(z) = 1/(1-4z) = \sum_{n=0}^{\infty} 4^n z^n$. Desta forma, $\tau_G(n) = 4^n$ o qual é igual ao número de sequências distintas definidas em Σ e de comprimento n quando não há ordem parcial.*

Para alguns grafos, pode ser complexo obter uma expressão fechada para $\tau_G(n)$, sendo necessário determinar uma expressão que se aproxime assintoticamente do valor de $\tau_G(n)$. O Teorema 2.2 é utilizado para garantir que seja possível determinar uma aproximação para o número de classes de equivalência de comprimento n, com um erro tendendo a zero para valores de n suficientemente grandes.

Teorema 2.2. *(Menor Raiz do Polinômio de Dependência [10]) O polinômio de dependência $D(G, z)$ tem uma raiz real de menor módulo z_0, tal que $0 < z_0 \leq 1$.*

2.4 Pilha de Peças

Como abordado anteriormente neste capítulo, as pilhas de peças são uma teoria equivalente aos monoides de comutatividade, mas com o apelo da interpretação geométrica [6]. Essa característica permitiu aplicação da teoria das pilhas em assuntos como mecânica estatística [11] e gravidade quântica [12]. As pilhas de peças utilizam extensivamente os conceitos de conjuntos parcialmente ordenados, os POSETs, e são melhor detalhados no Apêndice B.

Definição 2.6 (Pilha Rotulada)**.** *Seja $\mathcal{B}$ um conjunto de peças básicas e seja um grafo de comutatividade $G = (V, E)$ em que há uma bijeção $\lambda : V \to \mathcal{B}$. Uma pilha de peças rotuladas é uma tripla $(\mathcal{P}, \leq, \theta)$ em que $(\mathcal{P}, \leq)$ é um POSET e θ é um mapeamento $\theta : \mathcal{P} \to V$, de tal forma que*

(i) Para todo $r, s \in \mathcal{P}$ *se* $\theta(r)\theta(s) \not\equiv_G \theta(s)\theta(r)$*, então* $r \leq s$ *ou* $s \leq r$*.*

(ii) Para todo $r, s \in \mathcal{P}$ *de forma que* $r \leq s$*, então* $\theta(r)\theta(s) \not\equiv_G \theta(s)\theta(r)$*.*

Se $r \leq s$, então diz-se que s está sobre r ou que r está abaixo de s na pilha. Na condição (ii), é afirmado que se dois símbolos não comutam, então eles não estão no mesmo nível da pilha. Já na condição (ii), é afirmado que se um símbolo está sobre o outro na pilha, então ambos não comutam entre si. A condição (ii) da definição de pilha pode substituída ainda por (ii'):

(ii') Para todo $r, s \in \mathcal{P}$ e $r \leq s$, então existe uma sequência de peças $r = r_1 \leq \cdots \leq r_k = s, r_i \in \mathcal{P}$, tal que $\theta(r_i)\theta(r_{i+1}) \not\equiv_G \theta(r_{i+1})\theta(r_i)$ para todo $i, 1 \leq i < k$.

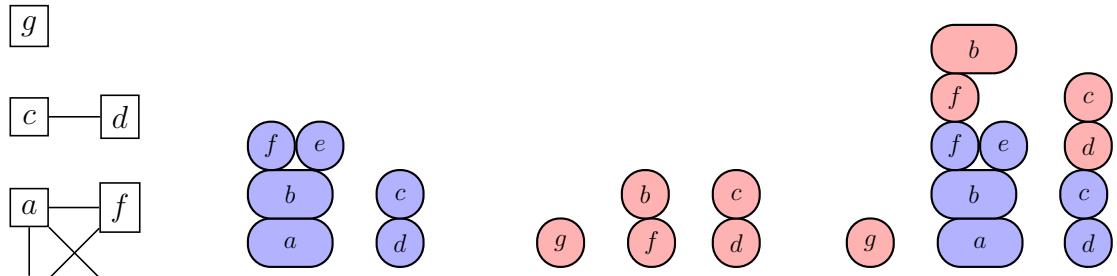
Da mesma forma que (ii), na condição (ii') é afirmado que se há um conjunto de peças empilhadas uma acima da outra, as peças adjacentes não comutam entre si. O conjunto de todas as pilhas finitas definidas com as peças de $\mathcal{B}$ e associado com um grafo de comutatividade G é representado por $\mathcal{H}(\mathcal{B}, G)$.

Definição 2.7 (Produto de Duas Pilha de Peças)**.** *Sejam as pilhas* $P = (\mathcal{P}, \leq_P, \theta_P)$ *e* $Q = (\mathcal{Q}, \leq_Q, \theta_Q)$ *definidas em* $\mathcal{H}(\mathcal{B}, G)$*. O produto entre* P *e* Q*, denotado por* $R = P \odot Q = (\mathcal{R}, \leq_R, \theta_R)$*, é definido como*

1. $\mathcal{R} = \mathcal{P} \cup \mathcal{Q}$*;*
2. *A ordem parcial* $\leq_R$ *em* R *é o fecho transitivo de*
   - *a)* $r \leq_R s$ *se* $r \leq_P s$*,*
   - *b)* $r \leq_R s$ *se* $r \leq_Q s$*,*
   - *c)* $r \leq_R s$ *se* $r \in \mathcal{P}$*,* $s \in \mathcal{Q}$ *e* $\theta_P(r)\theta_Q(s) \not\equiv_G \theta_P(s)\theta_Q(r)$*.*

A interpretação do produto de pilhas é empilhar as peças de uma pilha sobre a outra. A definição do produto de duas pilhas é associativo e $\mathcal{H}(\mathcal{B}, G)$ é chamado de monoide de pilha. Cabe notar que o produto de pilhas somente é comutativo se todos os elementos das pilhas forem comutativos.

Exemplo 2.6 (Produto de Pilha de Peças)**.** *Considere como exemplo o grafo de não - comutatividade da Fig. 2.5a e o conjunto de peças básicas* $\mathcal{B} = \{a, b, c, d, e, f, g\}$*, formando o conjunto de pilhas* $\mathcal{H}(\mathcal{B}, G)$*. Considere as pilhas das figuras 2.5b e 2.5c, que representam as palavras* $adbcfe$ *e* $fdgcb$*, respectivamente. O produto* $P \odot Q$ *é representado na Fig.2.5d e corresponde à palavra* $gadbcfedfcb$*.*



(a) $\overline{G}$. (b) Pilha P, representando $adbcfe$. (c) Pilha Q, representando $fdgcb$. (d) Pilha $P \odot Q$, representando $gadbcfedfcb$.

Figura 2.5 – Representação do produto de duas pilhas $P \odot Q$.

Tomando como exemplo a pilha representada na 2.5b, como a e d comutam de acordo com G, então elas estão no mesmo nível. Tratando de uma interpretação física de peças reais, olhando a pilha com a e d, não seria possível concluir qual das duas peças foi colocada primeiro. Entretanto, a e b não comutam e b está acima de a, pois a ocorreu primeiro. As palavras representadas pelas pilhas definem uma classe de equivalência de acordo com G, valendo também para o produto de pilhas. Essa relação será formalizada no decorrer desta seção.

As pilhas são formadas pelas chamadas pilhas triviais. Uma pilha trivial é aquela em que não existe peças sobre outras.

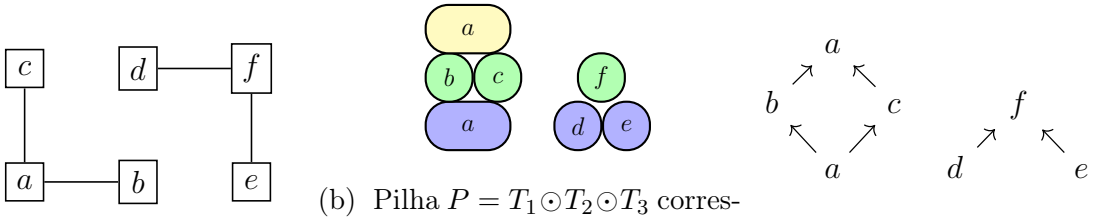
Definição 2.8 (Pilha Trivial)**.** *Uma pilha trivial é aquela em que a relação de ordem $\leq$ é trivial.*

As pilhas triviais da teoria das pilhas de peças correspondem aos cliques do grafo de comutatividade. O conjunto de pilhas triviais definidas com as peças de $\mathcal{B}$ com o grafo de comutatividade G é denotado por $\mathcal{T}(\mathcal{B}, G)$.

Lema 2.1 (Fatoração de uma Pilha)**.** *Qualquer pilha $P \in \mathcal{H}(\mathcal{B}, G)$ pode ser decomposta de uma forma única em pilhas triviais, $P = T_1 \odot \cdots \odot T_k$ de tal forma que qualquer peça de T_{i+1} está acima de uma peça de T_i, $1 \leq i < k$ e $T_i \in \mathcal{T}(\mathcal{B}, G)$.*

A fatoração descrita no Lema 2.1 pode ser obtida formando pilhas com os elementos mínimos da pilha P. No Exemplo 2.7, esse lema é explorado.

Exemplo 2.7 (Decomposição de Pilha de Peças)**.** *Considere como exemplo o grafo de não-comutatividade da Fig. 2.6a e conjunto de peças básicas $\mathcal{B} = \{a, b, c, d, e, f\}$, formando o conjunto de pilhas $\mathcal{H}(\mathcal{B}, G)$. Considere o diagrama de Hasse Fig. 2.6c representando a classe de equivalência da palavra $adefbca$ e a pilha da Fig. 2.6b.*



(a) Grafo de não-comutatividade

(b) Pilha $P = T_1 \odot T_2 \odot T_3$ correspondente à classe da palavra $adefbca$.

(c) Diagrama de Hasse associado à Pilha P.

Figura 2.6 – Grafo de não-comutatividade, pilha e POSET.

A pilha P pode ser decomposta nas pilhas triviais $P = T_1 \odot T_2 \odot T_3$, sendo T_1 formada por $\{a, d, e\}$, T_2 formada por $\{b, c, f\}$ e T_3 formada por $\{a\}$.

Definição 2.9 (Mapeamento de palavras em Pilhas)**.** *Seja o mapeamento $\phi : \mathcal{P}^* \to \mathcal{H}(\mathcal{B}, G)$ definido da seguinte maneira. Se $\mathbf{u} = u_1 u_2 \cdots u_n \in \mathcal{P}^*$, então $\phi(\mathbf{u}) = u_1 \odot u_2 \odot \cdots \odot u_n \in \mathcal{H}(\mathcal{B}, G)$. O mapeamento ϕ é o único morfismo de monoides tal que $u_i \in \mathcal{P}$ e $\phi(u_i)$ é uma pilha definida com uma peça básica u_i. Além disso, o inverso $\phi^{-1}(\mathbf{u})$ leva da pilha $\phi(\mathbf{u})$ para a palavra $\mathbf{u}$.*

Definição 2.10 (Rotulação Natural)**.** *Uma rotulação natural de um POSET $P = (\mathcal{P}, \leq_P)$ é uma bijeção $f : \mathcal{P} \to [n] = \{1, 2, \ldots, n\}$, tal que,' para todo $r, s \in \mathcal{P}$, se $r \leq_P t$, então $f(r) \leq f(s)$*

Uma rotulação natural é equivalente à definição de uma extensão linear de P e $\mathcal{L}(P)$ corresponde ao conjunto de extensões lineares de P (ver Apêndice B).

Lema 2.2. *Seja $(\mathcal{P}, \leq, \theta)$ uma pilha em $\mathcal{H}(\mathcal{B}, G)$ formada pelo POSET $P = (\mathcal{P}, \leq)$ e a função de rotulação θ. Considere $\mathbf{u} = u_1 u_2 \cdots u_n \in \phi^{-1}(\mathcal{P})$ e seja $\psi(\mathbf{u}) = f$ a rotulação $f : E \to [n]$ definida como $\theta(f^{-1}(i)) = u_i$. O mapeamento ψ é uma bijeção entre o conjunto de palavras $\phi^{-1}(\mathcal{P})$ e o conjunto de rotulações naturais é denotado por $\mathcal{L}(P)$.*

Lema 2.3. *Para cada pilha $P \in \mathcal{H}(\mathcal{B}, G)$, o conjunto de palavras $\phi^{-1}(\mathcal{P})$ corresponde a uma classe de equivalência definida pela relação de congruência $\equiv_G$.*

Os Lemas 2.2 e 2.3 relacionam o número de palavras de uma determinada classe ao número de extensões lineares do POSET associado, permitindo chegar à Proposição 2.1.

Proposição 2.1. *Seja $\mathcal{H}(\mathcal{B}, G)$ um monoide de pilha com peças pertencentes ao conjunto $\mathcal{B}$ e relações de comutatividade definidas por G. O morfismo do monoide $\phi : \mathcal{B}^* \to \mathcal{H}(\mathcal{B}, G)$ da Definição 2.9 induz um isomorfismo $\overline{\phi}$ entre o monoide $\mathcal{H}(\mathcal{B}, G)$ e o monoide de comutatividade $\mathcal{M}(\Sigma, G)$, ou ainda,*

$$\overline{\phi} : \mathcal{M}(\Sigma, G) \to \mathcal{H}(\mathcal{B}, G). \tag{2.5}$$

De fato, determinar o número de extensões lineares de um POSET equivale a determinar o número de palavras de determinada classe de equivalência. Desta forma, pode-se reescrever o Teorema B.3 (ver o Apêndice B) como:

Teorema 2.3. *Contar o número elementos de uma classe $\mathcal{E}$ é um problema #P-completo.*

A classe #P (lido como P-número) corresponde aos problemas da forma "determine o número de soluções corretas computadas em uma máquina de Turing não-determinística em um tempo polinomial". Um problema é #P-completo se pertence à classe #P e também à classe #P-difícil, o que significa que outros problemas da classe #P poderiam ser reduzidos a um problema #P-difícil. Isso torna os problemas #P-completos tão difíceis quanto os NP-Completos (ver Apêndice D).

Desta forma, determinar o número de palavras de uma classe também é um problema #P-completo e, portanto, difícil. Entretanto, a partir do POSET associado à pilha, é possível determinar limitantes para o número de extensões lineares e, portanto, para o número de palavras da classe de equivalência usando o Teorema B.4.

Seja uma classe de equivalência $\mathcal{E}_G(\mathbf{u}) \in \mathcal{M}(\Sigma, G)$. Para todas as palavras $\overline{\mathbf{u}} = u_1 u_2 \cdots u_n \in \mathcal{E}_G(\mathbf{u})$, a pilha $\phi(\overline{\mathbf{u}})$ correspondente é a mesma. Ou seja, todas as palavras de uma classe de equivalência correspondem à mesma pilha. Esse fato pode ser utilizado para enunciar o Corolário 2.2 que é semelhante ao Lema 2.1 acerca de fatoração de pilhas em pilhas triviais, mas em termos de monoides de comutatividade. Além disso, o número de palavras de uma classe é determinada pelo número de extensões lineares do POSET associado, como apresentado no Lema 2.2.

Corolário 2.2 (Forma Normal de Foata)**.** *Seja uma palavra $\mathbf{u} \in \mathcal{B}^*$ e o monoide de comutatividade $\mathcal{M}(\Sigma, G)$. Então, toda palavra $\overline{\mathbf{u}} \in \mathcal{E}_G(\mathbf{u})$ pode ser decomposta de forma única em $\overline{\mathbf{u}} = \mathbf{u}_1 \cdots \mathbf{u}_p$. Cada parcela $\mathbf{u}_i$ é um bloco de letras tais que*

1. *Todos os elementos de $\mathbf{u}_i$ comutam par a par; e*
2. *Para cada par de blocos consecutivos, $\mathbf{u}_i \mathbf{u}_{i+j}$, qualquer letra de $\mathbf{u}_i$ não comuta com pelo menos uma letra de $\mathbf{u}_{i+1}$.*

Desta forma, a decomposição é chamada de forma normal de Foata.

A forma normal de Foata (FNF) pode ser utilizada para provar algumas propriedades dos monoides de comutatividade e pilha de peças, assim como determinar uma certa medida de paralelismo de sistemas. Por exemplo, se a palavra $\overline{\mathbf{u}} \in \mathcal{E}_G(u)$ representa uma sequência de eventos de um determinado sistema, então cada parcela $\mathbf{u}_p$ de sua forma normal $\overline{\mathbf{u}} = \mathbf{u}_1 \cdots \mathbf{u}_p$ contém os eventos que podem ser executados de forma independente.

2.5 Resultados

Nessa seção, serão apresentadas as contribuições relacionadas à enumeração e obtenção de desigualdades para classes de equivalência. Também são apresentadas desigualdades relacionadas ao número de elementos das classes de equivalência. Esses resultados serão utilizados posteriormente para analisar algoritmos que fazem o uso de classes de equivalência.

2.5.1 Estimação do Número de Classes de Equivalência

Para o desenvolvimento de aproximações de $\tau_G(n)$, será definido o que é necessário para que duas funções sejam assintoticamente equivalentes. Diz-se que duas funções, $f(n)$ e $g(n)$, são assintoticamente equivalentes quando

$$f(n) \sim g(n) \iff \lim_{n \longrightarrow \infty} \frac{f(n)}{g(n)} = 1. \tag{2.6}$$

Para obter uma expressão assintoticamente equivalente a $\tau_G(n)$, deve-se realizar a expansão em frações parciais de 2.4, levando [13, cap.6] a

$$\begin{aligned} \zeta_M(z) &= \frac{1}{\sum_{k=0}^{\omega}(-1)^k c_k z^k} \\ &= \frac{\alpha_0}{(z-z_0)^{\rho_0}} + \frac{\alpha_1}{(z-z_1)^{\rho_1}} + \cdots + \frac{\alpha_\kappa}{(z-z_\kappa)^{\rho_\kappa}} \qquad (2.7) \\ &= \sum_{n=0}^{\infty} \tau_0(n) z^n + \sum_{n=0}^{\infty} \tau_1(n) z^n + \ldots + \sum_{n=0}^{\infty} \tau_\kappa(n) z^n, \qquad (2.8) \end{aligned}$$

em que $\alpha_\kappa \neq 0$ e $\sum_{n=0}^{\infty} \tau_i(n) z^n = \alpha_i (z - z_i)^{-\rho_i}$, $i = 0,1,\ldots,\kappa$.

Assumindo que z_0 é a raiz real de menor módulo de um polinômio de dependência $D(G,z)$, então $0 < z_0 < z_1 < \cdots < z_\kappa$. Desta forma, $\tau_G(n)$ pode ser expresso por um polinômio exponencial [13, cap.8] tal que

$$\tau_G(n) = \sum_{i=0}^{\kappa} \tau_i(n) = \sum_{i=0}^{\kappa} P_i(n) \left(\frac{1}{z_i}\right)^n, \tag{2.9}$$

em que $P_i(n)$ é uma função polinomial de n e que apresenta um coeficiente dominante igual a α. É possível, então, estimar $\tau_G(n)$ aplicando a equivalência assintótica [13, cap.8]

$$\tau_G(n) \sim \alpha n^{\deg(P_0(n))} \left(\frac{1}{z_0}\right)^n = \alpha n^{\deg(P_0(n))} \beta(G)^n, \tag{2.10}$$

em que $\deg(\cdot)$ é o grau do polinômio e $\beta(G)$ é o inverso da menor raiz de $D(G,z)$, $\beta(G) = \frac{1}{z_0}$.

Os termos $\alpha_i(z-z_i)^{-\rho_i}$, nos quais ρ_i é a multiplicidade da raiz z_i, podem ser expandidos por meio da série de Taylor, obtendo-se

$$\frac{\alpha_i}{(z-z_i)^{\rho_i}} = \sum_{n=0}^{\infty} \alpha_i (-1)^{\rho_i} \binom{n+\rho_i-1}{n} \left(\frac{1}{z_i}\right)^{n+\rho_i} z^n. \tag{2.11}$$

Usando (2.9) e (2.11), chega-se a

$$P_i(n) = \alpha_i \left(\frac{1}{z_i}\right)^{\rho_i} (-1)^{\rho_i} \binom{n+\rho_i-1}{n} \tag{2.12}$$

e, considerando a raiz z_0,

$$P_0(n) = \alpha_0 \beta(G)^{\rho_0} (-1)^{\rho_0} \binom{n+\rho_0-1}{n}. \tag{2.13}$$

Para obter o coeficiente dominante α de $P_0(n)$ e o seu grau $\deg(P_0(n))$, é necessário desenvolver o fator $\binom{n+\rho_0-1}{n}$. Nesse caso, tem-se

$$P_0(n) = \alpha_0 \beta(G)^{\rho_0} (-1)^{\rho_0} \left(\frac{(n+\rho_0-1)!}{(\rho_0-1)!n!}\right) \tag{2.14}$$

$$= \alpha_0 \beta(G)^{\rho_0} (-1)^{\rho_0} \left(\frac{(n+\rho_0-1)(n+\rho_0-2)\cdots(n+\rho_0-(\rho_0-1))n!}{(\rho_0-1)!n!}\right) \tag{2.15}$$

$$= \frac{\alpha_0 \beta(G)^{\rho_0} (-1)^{\rho_0}}{(\rho_0-1)!} \left((n+\rho_0-1)(n+\rho_0-2)\cdots(n+\rho_0-(\rho_0-1))\right) \tag{2.16}$$

Sabendo que $(n+\rho_0-1)(n+\rho_0-2)\cdots(n+\rho_0-(\rho_0-1))$ possui $\rho-1$ termos contendo n, conclui-se que $\deg(P_0(n)) = \rho_0 - 1$ e que o coeficiente dominante de $P_0(n)$ é $\alpha = \alpha_0 \beta(G)^{\rho_0}(-1)^{\rho_0}/(\rho_0-1)!$. Aplicando essa resultado em (2.10), obtém-se que

$$\tau_G(n) \sim \frac{\alpha_0(-1)^{\rho_0}}{(\rho_0-1)!} n^{(\rho_0-1)} \beta(G)^{n+\rho_0} \tag{2.17}$$

Para o caso particular de $\rho_0 = 1$, ou seja, quando a raiz z_0 tem multiplicidade igual a 1, tem-se que

$$\tau_G(n) \sim -\alpha_0 \beta(G)^{n+1}. \tag{2.18}$$

Cabe notar que ambas (2.17) e (2.18) podem ser determinadas a partir da decomposição em frações parciais de (2.7), mais particularmente do termo relacionado a raiz de menor módulo z_0. Isso fornece um modo rápido de aproximar o número de classes de intercâmbio de comprimento n.

O resultado em (2.18) é similar às conclusões apresentadas por Goldwurm e Santini [10]. No entanto, nesta dissertação mostra-se como determinar, a partir do grafo de comutatividade G, os valores das constantes usadas para se obter a aproximação de $\tau_G(n)$.

2.5.2 Limitantes para Número de Classes de Equivalência

Determinar os valores c_k do polinômio de dependência pode ser uma tarefa complicada em grandes grafos, pois o problema de encontrar os cliques de um grafo é NP-completo (ver Apêndice D). Para ainda poder utilizar limitantes mais justos que os triviais para o número $\tau_G(n)$ de classes de equivalência de comprimento n, é possível utilizar um limite superior para os valores de c_k dado no Teorema 2.4.

Teorema 2.4. *[14, Teorema 2, p.461] O número c_k de cliques de tamanho k considerando todos os grafos com $|E|$ arestas é limitado por*

$$c_k \leq f(|E|, k) = \binom{t}{k} + \binom{r}{k-1} \tag{2.19}$$

em que os valores de r e t são obtidos por meio de $|E| = \binom{t}{2} + r, 0 < r \leq t$.

Como o Teorema 2.4 não considera o número de vértices, nos casos de grafos em que o grau dos vértices é baixo, o limite superior se torna mais distante dos números de cliques de tamanho k observados, conforme o número de arestas cresce. O limitante se aproxima do valor correto de cliques conforme o número de arestas se aproxima de $\binom{|V|}{2}$.

Um grafo de $|V|$ vértices e $|E|$ arestas e pode possuir cliques de tamanho $k = 2, \ldots, \nu, \nu \leq |V|$, em que cada um desses subgrafos possui $|E_k| = \binom{k}{2}$ arestas. Desta forma, para um grafo poder possuir um clique de tamanho k, é necessário que $|E| \geq |E_k|$. Considere que o maior subgrafo possível de um grafo com $|E|$ arestas possui tamanho ν, então, de forma geral

$$|E| = \binom{\nu}{2} + r, 0 \leq r < \nu \text{ e } 2 \leq \nu \leq |V|. \tag{2.20}$$

Desta forma, para um grafo com $|V| = 5$ vértices e 5 arestas, tem-se $5 = \binom{3}{2} + 2$, então o maior clique possível teria tamanho $\nu = 3$. Se o número de arestas for 8, então $8 = \binom{4}{2} + 2$ e o maior clique possível terá tamanho $\nu = 4$.

O polinômio de dependência de uma família de grafos com $|E|$ arestas considerando o

limite superior para os valores de c_k será, portanto

$$\overline{D}(G,z) = 1 - |V| + |E|z^2 + \sum_{k=3}^{\nu}(-1)^k f(|E|,k)z^k \tag{2.21}$$

$$= 1 - |V| + |E|z^2 + \sum_{k=3}^{\nu}(-1)^k \overline{c}_k z^k. \tag{2.22}$$

Consequentemente, a função geradora do monoide de comutatividade será

$$\overline{\zeta}_M(z) = \frac{1}{1 - |V| + |E|z^2 + \sum_{k=3}^{\nu}(-1)^k f(|E|,k)z^k} = \tag{2.23}$$

$$= \frac{\alpha_0}{(z-z_0)^{\rho_0}} + \frac{\alpha_1}{(z-z_1)^{\rho_1}} + \cdots + \frac{\alpha_\kappa}{(z-z_\kappa)^{\rho_\kappa}} \tag{2.24}$$

$$= \sum_{n=0}^{\infty}\overline{\tau}_0(n)z^n + \sum_{n=0}^{\infty}\overline{\tau}_1(n)z^n + \ldots + \sum_{n=0}^{\infty}\overline{\tau}_\kappa(n)z^n. \tag{2.25}$$

A partir de (2.23), é possível aplicar as técnicas de estimação desenvolvidas na Seção 2.5.1. Para exemplificar a utilização dos limitantes, serão apresentadas algumas aplicações em grafos com um número de vértices $|V| \geq 10$.

Exemplo 2.8. *Considerando o grafo de comutatividade G_1 da Fig. 2.7, gerado aleatoriamente com $|V| = 10$ e $|E| = 32$, tem-se que $c_3 = 42, c_4 = 23, c_5 = 4$. Desta forma, o número do clique é $\omega = 5$ e $\nu = 8$.*

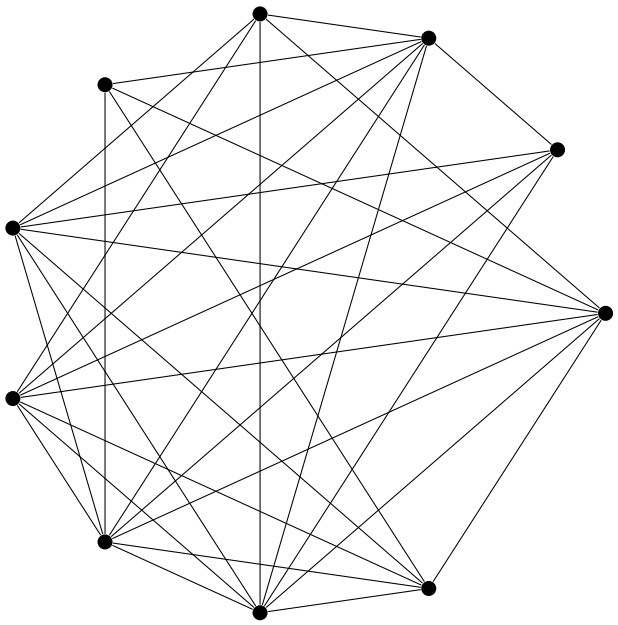


Figura 2.7 – Grafo de comutatividade G_1 com $|V| = 10, |E| = 32, c_3 = 42, c_4 = 23, c_5 = 4$.

O polinômio de dependência do grafo da Fig. 2.7 será

$$D(G_1,z) = 1 - 10z + 32z^2 - 42z^3 + 23z^4 - 4z^5 \tag{2.26}$$

E o polinômio gerador será

$$\begin{aligned}\zeta_{\mathcal{M}}(z) &= \frac{1}{1-10z+32z^2-42z^3+23z^4-4z^5} \\ &\sim -\frac{0{,}54380}{z-0{,}18785}.\end{aligned} \tag{2.27}$$

Aplicando 2.18, chega-se a

$$\tau_{G_1}(n) \sim 0{,}54380\left(\frac{1}{0{,}18785}\right)^{n+1} \tag{2.28}$$

$$\approx 2{,}8948 \cdot 5{,}3234^n \tag{2.29}$$

Agora usando o Teorema 2.4 e a (2.20), pode-se obter o polinômio de dependência da família de grafos ao qual G pertence como

$$\overline{D}(G_1, z) = \sum_{k=0}^{\kappa}(-1)^k\overline{c}_k z^k = 1-10z+32z^2-62z^3+74z^4-57z^5+28z^6-8z^7+1z^8 \tag{2.30}$$

correspondendo ao polinômio gerador $\overline{\zeta}_M$ *dado por*

$$\overline{\zeta}_{\mathcal{M}}(z) \sim -\frac{0{,}2920}{z-0{,}1624}$$

Aplicando (2.18) mais uma vez, conclui-se que o limitante para o número de classes de equivalência da família de grafos G_1 *será*

$$\overline{\tau}_{G_1}(n) \sim 0{,}2920 \cdot 6{,}1584^n \tag{2.31}$$

Na Fig. 2.8 é apresentado um gráfico contendo o limite trivial $|\Sigma|^n$*, e as estimações obtidas. Cabe notar que o limite é mais justo que o trivial.*

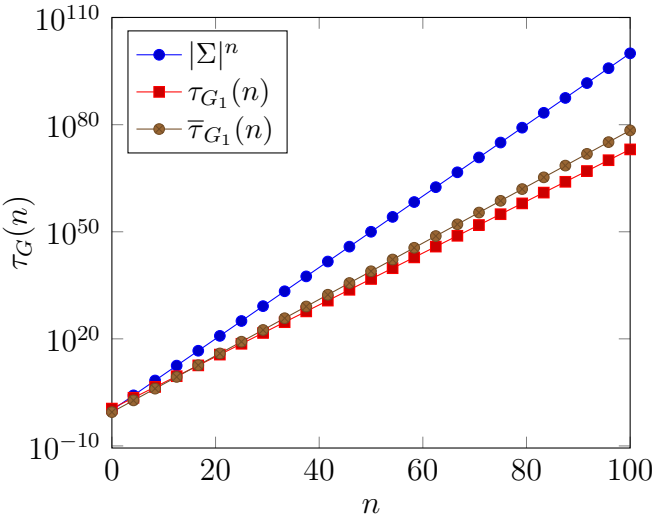


Figura 2.8 – Comparação entre o número de classes de equivalência e o limite superior obtido para G_1.

Exemplo 2.9. *Para esse exemplo, será considerado o grafo de comutatividade da Fig. 2.9 que possui* $|V| = 25$ *vértices e* $|E| = 80$ *arestas.*

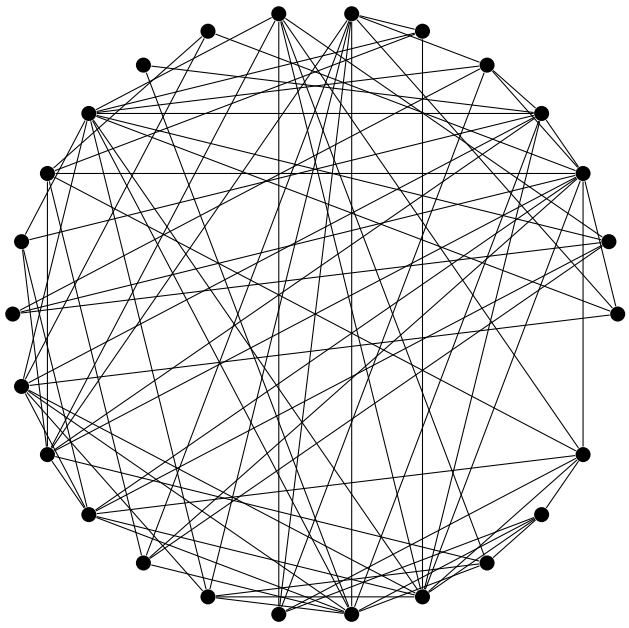


Figura 2.9 – Grafo de comutatividade G_2 com $|V| = 25$, $|E| = 80$, $c_3 = 46$, $c_4 = 5$.

O polinômio de dependência será

$$D(G_2,z) = 1 - 25z + 80z^2 - 46z^3 + 6z^4$$

e, seguindo o mesmo procedimento do exemplo anterior, obtém-se

$$\tau_{G_2}(n) \sim 0{,}0562 \cdot 21{,}3539^n. \tag{2.32}$$

Como esse grafo contém 25 vértices e 80 arestas, então usando (2.20), tem-se

$$80 = \binom{13}{2} + 2 \implies \nu = 13.$$

Usando agora o Teorema 2.4, tem-se que $c_3 \leq 287.0, c_4 \leq 715.0, c_5 \leq 1287.0, c_6 \leq 1716.0, c_7 \leq 1716.0, c_8 \leq 1287.0, c_9 \leq 715.0, c_{10} \leq 286.0, c_{11} \leq 78.0, c_{12} \leq 13.0$ *e* $c_{13} \leq 1.0$.

Dessa maneira, o polinômio de dependência, nesse caso, será

$$\begin{aligned}\overline{D}(G_2, z) = 1 - 25z + 80z^2 - 287z^3 + 715z^4 - 1287z^5 + 1716z^6 - \\ - 1716z^7 + 1287z^8 - 715z^9 + 286z^{10} - 78z^{11} + 13z^{12} - 1z^{13},\end{aligned}$$

obtendo-se que

$$\overline{\tau}_{G_2}(n) \sim 0{,}0520 \cdot 21{,}8802^n$$

Na Fig. 2.10 é apresentado um gráfico contendo o limite trivial $|\Sigma|^n$*, e as estimações obtidas. Cabe notar que o limite também é bem mais justo que o trivial nesse caso.*

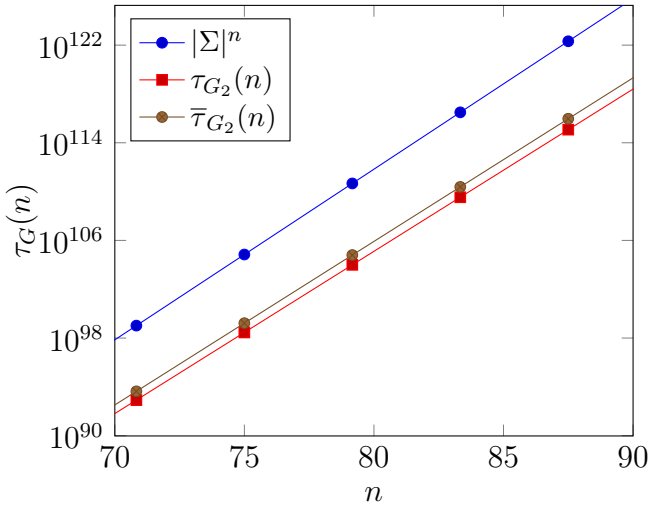


Figura 2.10 – Comparação entre o número de classes de equivalência e o limite superior obtido para G_2.

2.5.3 Limitantes para o Número de Palavras de uma Classe de Equivalência

A partir do Lema 2.3 é possível reescrever o Teorema B.4, que inicialmente é utilizado para determinar limitantes inferior e superior para o número de extensões lineares de um POSET P, a fim de obter limitantes para o número de elementos de uma classe de equivalência.

Teorema 2.5 (Número de elementos de uma classe de equivalência)**.** *Seja uma palavra* $\mathbf{u} \in \mathcal{E}_G(\mathbf{u})$ *e* $\phi(\mathbf{u}) \in \mathcal{H}(\mathcal{P}, G)$ *a pilha correspondente,* $(\mathcal{P}, \leq, \theta)$*. Fazendo a decomposição de* $(\mathcal{P}, \leq)$ *em* k *cadeias* $C_1, \ldots, C_k$ *e* l *anti-cadeias* $A_1, \ldots, A_l$ *e, assumindo que* $n = |\mathbf{u}|$*, então o número de palavras da classe* $\mathcal{E}_G(\mathbf{u})$ *é limitado por*

$$\prod a_i! \leq |\mathcal{E}_G(\mathbf{u})| \leq \frac{n!}{\prod c_i!}. \tag{2.33}$$

Prova do Teorema 2.5. Segue imediatamente do Teorema B.4 e do Lema 2.3. □

Há ainda duas questões de ordem prática relacionada à aplicação do Teorema 2.5. A primeira é que para obter os limitantes, é necessário decompor o POSET em cadeias e anti-cadeias maiores possíveis, o que não é um problema trivial. A segunda é se os limites obtidos são úteis no sentido de serem justos o suficiente. Para resolver a primeira, é possível utilizar o Algoritmo de Ford-Fulkerson [15](ver Apêndice B), como já feito em [16]. Já a segunda foi abordada em [17], no qual foi demonstrado que os limitantes obtidos são justos.

Em muitos casos, uma estimação do número de classes é suficiente. Por exemplo, o Teorema 2.5 pode ser utilizado para estimar a complexidade temporal e espacial de algoritmos que geram a classe de equivalência de uma palavra. Esses algoritmos são

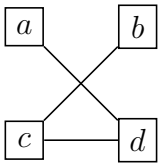
de interesse pois possibilitam a resolução de problemas de agendamento (*scheduling*), otimização, ordenação topológica, dentre outros [18, 19, 20, 21, 22].

Para exemplificar a aplicação do Teorema 2.5, será considerado o Algoritmo 1. Esse algoritmo é uma versão ingênua do algoritmo proposto em [22] e possui uma complexidade temporal $O(n|\mathcal{E}(\mathbf{u})|)$, em que $\mathbf{u}$ é a palavra de entrada. É possível transformá-lo em um algoritmo com complexidade $O(|\mathcal{E}(\mathbf{u})|)$ ao manter os índices dos pares de símbolos que comutam armazenados, ao invés de procurá-los ao percorrer a sequência. Por meio da estimação, é possível decidir, por exemplo, se é factível gerar todos os elementos de uma classe de equivalência. Entretanto, para as análises realizadas aqui, o algoritmo apresentado é suficiente.

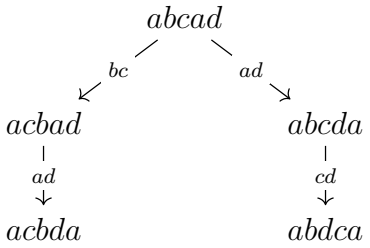
Algoritmo 1: Geração de Classes Equivalentes

```
1  procedimento GeraEquivalentes(u, G, H):
2      E ← ∅
3      para i ← 1 até i = |u| faça
4          se (u_i, u_{i+1}) ∈ E então
5              v ← u_1 ··· u_{i+1} u_i ··· u_n
6              se v ∉ H então
7                  E ← E ∪ v
8                  H[v] = v                    ▷ Adiciona nova palavra na tabela
9      retorna E                               ▷ Palavras congruentes a u
10 procedimento GeraClasse(u, G):
11     H ← Cria Tabela Hash
12     N ← GeraEquivalentes(u, G)              ▷ Palavras equivalentes a u por uma troca
13     enquanto |N| ≠ 0 faça
14         M ← N
15         N ← ∅
16         para n ∈ M faça
17             N ← N∪ GeraEquivalentes(n, G, H)
18     retorna H                               ▷ Tabela hash contendo os elementos da classe de u
```

Na Fig. 2.11, são apresentados um grafo de comutatividade (Fig. 2.11a) e a árvore formada ao aplicar o Algoritmo 1 na palavra $\mathbf{u} = abcad$. A partir de $\mathbf{u}$ são identificados dois pares de símbolos que comutam, bc e ad, gerando $acbad$ e $abcda$, respectivamente. O processo é repetido para as novas palavras, até os pares de símbolos comutativos não gerarem novas palavras.



(a) Grafo de comutatividade G.



(b) Árvore das palavras equivalentes a *abcad*.

Figura 2.11 – Exemplo de aplicação do Algoritmo 1.

Exemplo 2.10. *Considera-se, como exemplo, um sistema concorrente que pode executar tarefas enumeradas de 1 a 9. As relações de ocorrência são apresentados pelo POSET representado pela Fig. 2.12. Desta forma, inicialmente, as tarefas 1 e 2 podem ser executadas. Se a tarefa 1 foi executada, a tarefa 3 pode ser executada. A tarefa 5 pode ser executada se 2 já foi executada, enquanto a 4 depende de 1 e 2, e assim por diante. Pode-se notar que tarefas independentes formam as anti-cadeias do POSET, enquanto as dependentes formam as cadeias.*

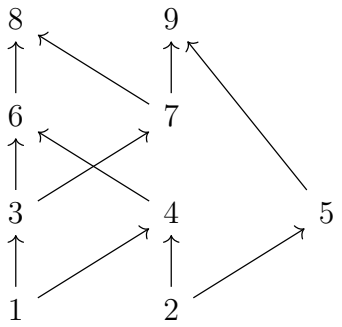


Figura 2.12 – Diagrama de Hasse de P.

Supondo que $\mathbf{u} = 125347689$ *seja a sequência de execução das tarefas, palavras da classe* $\mathcal{E}(\mathbf{u})$ *produzirão o mesmo resultado de* $\mathbf{u}$. *Deseja-se saber de quantas maneiras pode-se escolher a ordem de execução das tarefas para que sejam equivalentes a* $\mathbf{u}$*, ou seja,* $|\mathcal{E}(\mathbf{u})|$. *Utilizando o Algoritmo de Ford-Fulkerson [15], é possível obter os tamanhos das cadeias do POSET representado na Fig. 2.12 como* $c_1 = |\{2{,}4{,}6{,}8\}| = 4$, $c_2 = |\{1, 3, 7\}| = 3$ *e* $c_3 = |\{5, 9\}| = 2$. *Os tamanhos das anti-cadeias podem ser obtidas por meio do Algoritmo 4 (ver Apêndice B), como* $a_1 = |\{1{,}2\}| = 2$, $a_2 = |\{3{,}4{,}5\}| = 3, a_3 = \{6{,}7\} = 2, a_4 = \{8{,}9\} = 2$. *Aplicando agora o Teorema 2.5, pode-se obter*

$$2!3!2!2! \leq |\mathcal{E}(\mathbf{u})| \leq \frac{9!}{4!3!2!} \implies 48 \leq |\mathcal{E}(\mathbf{u})| \leq 1260. \tag{2.34}$$

Aplicando o Algoritmo 1, chega-se a que $|\mathcal{E}(\mathbf{u})| = 199$ *palavras fazem parte da classe definida por* $\mathbf{u}$. *Esse valor é cerca de 6 vezes menor que o limite superior obtido, mas,*

ainda assim, é bem menor do que todas as 9! *permutações, caso não houvesse ordem parcial.*

2.6 Conclusões

Nesse capítulo, foram apresentados os conceitos fundamentais dos monoides de comutatividade e pilha de peças. O conceito de grafo comutatividade é fundamental para formalizar o tratamento de alfabetos com ordem parcial. Essa formalização permite contar o número de classes de comprimento n. Entretanto, em alguns casos se torna difícil obter uma equação simples para descrever como o número de classes cresce de acordo com o seu comprimento. Além disso, a enumeração das classes depende da contagem dos números de cliques do grafo de comutatividade, o que é problema difícil computacionalmente.

Como resultados deste capítulo, foi desenvolvido um modo de estimar o número de classes de equivalência de comprimento n a partir do grafo de comutatividade. A estimação permite a obtenção de uma equação simples e assintoticamente equivalente ao número de classes de equivalência. Foram também obtidos limitantes para o número de classes de equivalência para famílias de grafos com $|V|$ vértices e $|E|$ arestas, o que pode ser utilizado quando contar os cliques for intratável computacionalmente. Por último, foi utilizado a teoria das pilha de peças para a obtenção de limitantes para o número de elementos pertencentes a uma mesma classe de equivalência.

No próximo capítulo serão abordadas as definições de entropias relacionadas aos alfabetos parcialmente comutativos e será proposta uma nova medida de entropia.

Capítulo 3

Entropia de Comutatividade

3.1 Introdução

Considerando um alfabeto Σ que é iid e com ordem total, a entropia de Shannon (ver Apêndice C) tem o limitante

$$\begin{aligned} H(\Sigma) &= -\sum_{\sigma\in\Sigma} p(\sigma)\log p(\sigma) \\ &= -\sum_{\sigma\in\Sigma} \frac{1}{|\Sigma|}\log\frac{1}{|\Sigma|} \\ &\leq \log|\Sigma|. \end{aligned} \tag{3.1}$$

Usando as terminologias dos monoides de comutatividade (ver Cap. 2), nesse caso, o grafo de comutatividade será vazio (não há nenhuma comutação entre os símbolos) e a função geradora do monoide será

$$\zeta_{\mathcal{M}}(z) = \sum_{n=0}^{\infty}\tau(n)z^n = \sum_{n=0}^{\infty}|\Sigma|^n z^n, \tag{3.2}$$

ou seja, o número de palavras de comprimento n é dado por $\tau(n) = |\Sigma|^n$ e o limitante apresentado em (3.1) pode ser escrito como $\log|\Sigma| = (1/n)\log\tau(n)$.

De forma geral, a função geradora do monoide também tem a forma $\zeta_{\mathcal{M}}(z) = \sum_{n=0}^{\infty}\tau(n)z^n$, em que $\tau(n) \leq |\Sigma|^n$ depende do grafo de comutatividade, sendo representado por $\tau_G(n)$. Desta forma, é esperado que a entropia do monoide tenha uma desigualdade semelhante ao caso com ordem total, mas fazendo algumas considerações. Para obter o limite superior da entropia de Shannon, o alfabeto deve ser iid, o que corresponde, no caso dos monoides, às classes de equivalência serem iid. Nesse caso, é esperado que a

entropia de comutatividade respeite a desigualdade

$$H(\Sigma, G) \leq \lim_{n\to\infty} \frac{1}{n} \log \tau_G(n), \tag{3.3}$$

Assim, o espaço amostral considerado passa a ser as classes de comprimento n e é necessário uma medida de probabilidade para essas classes. Quando G é um grafo de comutatividade vazio, ou seja, representa uma ordem total, o limitante de (3.3) é igual ao limitante da entropia de Shannon em 3.1.

Exemplo 3.1. *Considere $\Sigma = \{a, b, c\}$ e o grafo de comutatividade da Fig. 3.1. Considere também que os símbolos são equiprováveis, $p(a) = p(b) = p(c) = 1/3$.*

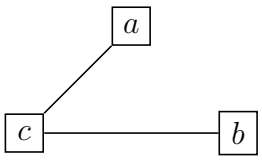


Figura 3.1 – Grafo de comutatividade G.

Nesse caso, a entropia de Shannon será

$$H(\Sigma) = \log 3 \approx 1.58\, bit. \tag{3.4}$$

Para calcular $\tau_G(n)$, primeiro determina-se o polinômio de dependência de G como $D(G, z) = 1 - 3z + 2z^2$ e função geradora como $1/D(G, z)$, obtendo-se $\tau_G(n) = 2^{n+1} - 1$. Pode-se calcular o limite superior de $H(\Sigma, G)$ como

$$H(\Sigma, G) \leq \lim_{n\to\infty} \frac{1}{n} \log(2^{n+1} - 1) \tag{3.5}$$

$$= \log 2 = 1\, bit \tag{3.6}$$

Na próxima seção, serão abordadas as principais definições de entropia que consideram alfabetos com ordem parcial. Em seguida, será a apresentada a definição de entropia proposta nessa dissertação.

3.2 Entropias de Alfabetos com Ordem Parcial

Nesta seção serão apresentadas entropias que podem ser aplicadas a alfabetos com ordem parcial. Dentre as apresentadas, duas foram definidas com o objetivo de serem aplicadas nesses alfabetos. A apresentação das entropias será feita em ordem cronológica.

3.2.1 Entropia Topológica

Seja um alfabeto Σ e o monoide livre Σ^*, uma linguagem L é qualquer subconjunto de Σ^*. Considere também a função $f_L(n)$ que retorna o número de elementos de comprimento n da linguagem L, $f_L(n) = |\{\mathbf{w} : \mathbf{w} \in L \wedge |\mathbf{w}| = n\}|$. Com esses conceitos, é possível definir a entropia de uma linguagem ou entropia topológica.

Definição 3.1 (Entropia Topológica [23])**.** *A entropia de uma linguagem L é definida como*

$$H(L) = \lim_{n\to\infty} \sup \frac{\log\left(f_L(n)\right)}{n}. \tag{3.7}$$

Essa entropia pode ser interpretada como quantidade de bits que são necessários em média para representar cada uma das classes de palavras de comprimento n pertencentes à L, conforme o comprimento das palavras aumenta.

A entropia topológica pode ser facilmente adaptada para uma entropia do monoide de comutatividade como:

Definição 3.2 (Entropia de Topológica do Monoide de Comutatividade)**.** *A entropia da linguagem L associada ao monoide de comutatividade $\mathcal{M}(\Sigma, G)$ é definida como*

$$H(G) = \lim_{n\to\infty} \sup \frac{\log(\tau_G(n))}{n}. \tag{3.8}$$

Apesar de ter propriedades interessantes e se aproximar da definição de uma taxa de entropia, não leva em consideração a probabilidade de cada uma das classes de palavras. Outra interpretação é que leva em consideração a probabilidade, mas que as classes são equiprováveis, de forma que o lado direito de (3.3) corresponde à definição da entropia topológica.

3.2.2 Entropia de Traços Infinitos

Manohar [24] propôs uma entropia baseada na distribuição de conjuntos de operações executadas em circuitos com integração em larga escala, VLSI (do inglês, *Very Large Scale Integration*). Isso foi motivado pela relação entre a entropia e a energia necessária para executar os conjuntos de operações. Se $E(S)$ é a energia dissipada pelo circuito para a especificação S e $H(S)$ é a entropia também considerando S, então $H(S) \leq E(S) \leq H(S) + K(S)$, em que $K(S)$ é um valor que depende do número de iterações dos cálculos.

Para definir a entropia dos traços, Manohar [24] começa com a definição do problema de codificação, a fim de obter um esquema ótimo de codificação.

Definição 3.3 (Problema de Codificação)**.** *Seja $\mathcal{U}$ um conjunto fixado com cardinalidade N e S um subconjunto arbitrário de $\mathcal{U}$ com cardinalidade k. Quantos bits de informação são necessários para identificar um elemento de S?*

A partir da definição do problema, Manohar [24] apresentou o Lema 3.1 que enuncia os limitantes inferior e superior para o número de bits de informação necessários para codificar um subconjunto.

Lema 3.1 (Limitantes para o número de bits)**.** *Dado um conjunto $\mathcal{U}$ com cardinalidade N e qualquer subconjunto $S \subseteq \mathcal{U}$ com cardinalidade $k > 0$, a informação necessária para identificar qualquer elemento de S se encontra entre* $\log(N-k+1)$ *e* $\lceil \log(N-k+1) \rceil$.

Por se tratar de circuitos VLSI, Manohar [24] assume que os traços são infinitos e propõe que os elementos de S podem ser escolhidos de forma independente. Nesse caso, se $k_i, k_j \in S$, então $\Pr(k_i k_j) = \Pr(k_i)\Pr(k_j)$. Desta forma, as probabilidades $p_{k_i} = \Pr(k_i)$ definem a distribuição $\Pr(\cdot)$.

Teorema 3.1 ([24])**.** *Assume-se que $\mathcal{U}$ é infinito e que os elementos de $S \subseteq \mathcal{U}$ são selecionados de forma independente. Seja $\{k_1, k_2, \dots\}$ o subconjunto de $\mathcal{U}$ que otimiza a codificação. Seja também q_{k_i} a probabilidade de selecionar k_i, em que, sem perda de generalidade, $q_{k_i} \geq q_{k_j}$ para $i \leq j$. Então, k_i é o i-ésimo elemento mais provável de $\mathcal{U}$ e o esquema ótimo de codificação corresponde à seleção de $k_i \in S$, de forma que $k_j \notin S$ para $i = 1,2,\cdots,i-1$, levando a*

$$q_{k_i} = p_{k_i} \prod_{i<j} \left(1 - p_{k_j}\right). \tag{3.9}$$

O Teorema 3.1 parte da simplificação de que os elementos do k-subconjunto são selecionados de forma independente, ou seja, que $\Pr(ab) = \Pr(a)\Pr(b)$. Isso pode ser assumido porque são considerados traços infinitos. A entropia da distribuição q_{k_i} é dada por

$$\sum_i \left(\prod_{i<j} \left(1 - p_{k_j}\right) \right) H(p_{k_i}), \tag{3.10}$$

em que $H(p_{k_i})$ é a entropia de Shannon considerando um alfabeto binário, $H(p_{k_i}) = -p_{k_i} \log p_{k_i} - (1 - p_{k_i}) \log (1 - p_{k_i})$.

O primeiro problema da abordagem de Manohar [24] é considerar os traços infinitos com o objetivo de selecionar os elementos de S de forma independente e, consequentemente, obter (3.9). Na maioria dos casos, os sistemas executam conjuntos de tarefas com um número finito de operações. Desta forma, os traços não seriam, necessariamente, independentes. O segundo problema é que não há um modo explícito de atribuir probabilidades aos traços em S. Isso se torna ainda mais sensível quando se considera que são traços infinitos. Desta forma, apesar de fornecer uma expressão para a entropia dos traços, a aplicação se torna difícil pelos fatores apresentados.

3.2.3 Entropia de Intercâmbio

Savari [25] propôs a entropia de intercâmbio que se baseia na complexidade de intercâmbio que é uma generalização da complexidade de Kolmogorov. Essa abordagem tem o paralelo no caso dos alfabetos com ordem total, em que é possível relacionar a entropia de Shannon com a complexidade de Kolmogorov (ver Teorema D.2), evitando problemas como os observados na Seção 3.2.2. A entropia de intercâmbio faz uso de ferramentas da teoria dos traços [2] para definir o limite teórico de compressão de sequências de eventos com ordem parcial em termos de números de *bits* por símbolo em média para representar as sequências [26]. Esses limites podem ser utilizados para determinar a eficiência de esquemas de codificação e compressão de sequências de eventos com ordem parcial. Além disso, o limite superior da entropia de intercâmbio é a entropia de Shannon e, portanto, quando se considera a ordem parcial, a compressão obtida é maior ou igual a conseguida com ordem total. Segundo Alur [27], até a publicação do artigo [26], também de Savari, não havia sinal de trabalhos focados no aspecto de teoria de distorção para sequências de eventos com ordem parcial.

Seja $K(\mathbf{u})$ a complexidade de Kolmogorov (ver Apêndice D) da palavra $\mathbf{u} \in \Sigma^*$, G um grafo de comutatividade e $\equiv_G$ a relação de congruência representada por G. A *Complexidade de Intercâmbio* é definida como $C_\iota(G, \mathbf{u}) = \min\{K(\mathbf{v}) : \mathbf{v} \in \Sigma^*, \mathbf{v} \equiv_G \mathbf{u}\}$. Assim, a complexidade de intercâmbio pode ser interpretada como o comprimento do menor programa que exibe a palavra $\mathbf{v} \in \Sigma^*$ que é congruente a $\mathbf{u}$ em relação ao grafo G.

A partir da complexidade de intercâmbio, Savari [25] também definiu a Propriedade de Equipartição Assintótica para Classes de Intercâmbio.

Teorema 3.2. *(Propriedade de Equipartição Assintótica para Classes de Intercâmbio, Teorema 2.5 de [25, p.1428]) Seja $U_1U_2\ldots$ uma palavra aleatória de uma fonte com alfabeto finito, estacionária e ergódica ou a saída de uma fonte unifilar de Markov com alfabeto finito. Então,*

$$\begin{aligned}\lim_{n\to\infty} \frac{C_\iota(G, U_1U_2\ldots U_n)}{n} &= \lim_{n\to\infty} \frac{\mathbb{E}\left[\{C_\iota(G, U_1U_2\ldots U_n)\}\right]}{n} \\ &= \inf_{m\geq 1} \frac{\mathbb{E}\left[\{C_\iota(G, U_1U_2\ldots U_m)\}\right]}{m}\end{aligned}$$

com probabilidade tendendo a 1.

O Teorema 3.2 é uma generalização do Teorema C.1 para alfabetos em que há uma ordenação parcial, sendo possível verificar que existem sequências chamadas de típicas. Essas sequências típicas apresentam uma informação própria média por símbolo próxima da entropia da fonte. Enquanto as sequências típicas ocorrem com alta probabilidade,

as sequências não-típicas acontecem com baixa probabilidade. Isso leva à definição da entropia de intercâmbio.

Definição 3.4 (Entropia de Intercâmbio)**.**

$$H_\iota(G,P) = \lim_{n\to\infty} n^{-1} C_\iota\left(G, U_1 \dots U_n\right). \tag{3.11}$$

em que P se refere a uma medida de probabilidade associada aos vértices de G. Há a convergência com alta probabilidade do limite para o valor da entropia de intercâmbio.

O Teorema 3.2 estabelece que, para qualquer $\epsilon > 0$, sequências longas requerem, no máximo, $H_\iota(G,P)+\epsilon$ *bits* por símbolo para descrever uma palavra equivalente em relação ao grafo G. O Teorema 3.3 estabelece que $H_\iota(G,P)$ é o limite de compressão.

Teorema 3.3. *(Teorema da Codificação, Teorema 2.6 [25, p.1429]) Seja $U_1U_2\dots$ uma sequência aleatória de saída de uma fonte P com estados finitos ou com uma fonte unifilar de Markov com alfabeto finito. Para $n \geq 1$, seja B_n o conjunto com cardinalidade $\tau_G(n) = |B_n|$ de classes de intercâmbio de comprimento n em relação a um grafo de comutatividade G. Assume-se que*

$$\limsup_{n\to\infty} \frac{\log|B_n|}{n} \leq H_\iota(G,P)$$

Para qualquer palavra w, $\mathcal{E}_G(\mathbf{w})$ denota a classe de equivalência que contém a palavra $\mathbf{w}$ em relação ao grafo de comutatividade $G = (V,E)$, $\mathbf{w} \in \Sigma^*$, que é definida como

$$\mathcal{E}_G(w) = \{\mathbf{y} : \mathbf{y} \in \Sigma^*,\, \mathbf{y} \equiv_G \mathbf{w}\}$$

Desta forma,

$$\lim_{n\to\infty} P(\mathcal{E}_G(U_1U_2\dots U_n) \in B_n) = 0.$$

O Teorema 3.3 é o paralelo do Teorema C.3 que afirma que é possível representar as sequências $X_1,\dots,X_n$, para X pertencente a um alfabeto finito $\mathcal{X}$, usando $nH(X)$ *bits* em média. É possível concluir que sequências típicas de comprimento n estão contidas em aproximadamente $2^{nH_\iota(G,P)}$ classes de intercâmbio típicas.

Classes de intercâmbio se assemelham ao conceito de classes de tipo (ver Definição C.5) e configuram uma generalização desse último. Cabe notar que no caso de G ser um grafo completo o número de classes de tipo de comprimento n para um conjunto V de vértices com cardinalidade $|V|$ é, no máximo, $(n+1)^{|V|}$. Esse é o número de classes de tipo convencionais, pois todos os símbolos são congruentes e podem ser comutados.

É possível obter uma expressão exata para $H_\iota(G,P)$ no caso da fonte ser discreta e sem memória e o grafo $\overline{G}$ ser um grafo k-partido completo $K_{m_1,m_2,\dots,m_k}$. Esse resultado é apresentado no Teorema 3.4.

Teorema 3.4. *([25, p.1432]) Assumindo uma fonte discreta e sem memória com distribuição P de probabilidade no conjunto de vértices V. Supondo que $V = V_1 \cup V_2 \cup \ldots \cup V_k$ com $|V_i| = m_i$, $i \in \{1, 2, \ldots, k\}$. O j-ésimo elemento de V_i será denotado por $v_{i,j}$, $i \in \{1, 2, \ldots, k\}$, $j \in \{1, 2, \ldots, m_i\}$. Para $i \in \{1, 2, \ldots, k\}$, define-se*

$$Q_i = \sum_{j=1}^{m_1} P(v_{i,j})$$

$$\varphi_i(S) = \left(Q_i^S - \sum_{j=1}^{m_i} \left(\frac{P(v_{i,j})}{1 - Q_i + P(v_{i,j})} \right)^S \right) \log(S)$$

Então, para um grafo k-partido completo $K_{m_1,m_2,\ldots,m_k}$, tem-se

$$H_\iota(K_{m_1,m_2,\ldots,m_k}, P) = H(P) - \sum_{S=2}^{\infty} \sum_{i:m_i \geq 2} (1 - Q_i)\varphi_i(S).$$

Exemplo 3.2. *Considerando o grafo de não-comutatividade $\overline{G}$ apresentado na Fig. 3.2, é possível verificar que ele é um grafo bipartido completo, já que $V = V_1 \cup V_2 = \{a, c\} \cup \{b\}, \{a, c\} \cap \{b\} = \emptyset$ e todos os elementos da partição V_1 são conectados por uma aresta a todos os outros elementos da partição V_2.*

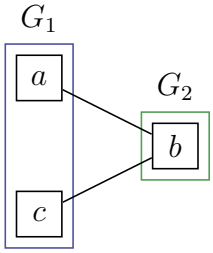


Figura 3.2 – Grafo $\overline{G} = K_{2,1}$ bipartido completo.

Desta forma, $V_1 = \{v_{1,1} = a, v_{1,2} = c\}$, $|V_1| = m_1 = 2$, $V_2 = \{v_{2,1} = b\}$, $|V_2| = m_2 = 1$, $Q_1 = P(a) + P(c) = 2/3$ e $Q_2 = P(b) = 1/3$.

$$\begin{aligned}
\varphi_1(S) &= \left(Q_1^S - \sum_{j=1}^{m_1} \left(\frac{P(v_{1,j})}{1 - Q_1 + P(v_{1,j})} \right)^S \right) \log(S) = \\
&= \left((2/3)^S - \left(\frac{1/3}{1 - 2/3 + 1/3} \right)^S - \left(\frac{1/3}{1 - 2/3 + 3} \right)^S \right) \log(S) \\
&= \left((2/3)^S - 2(1/2)^S \right) \log(S).
\end{aligned}$$

Assim, calculando $H_\iota(K_{2,1}, P)$*, tem-se*

$$\begin{aligned} H_\iota(K_{2,1}, P) &= H(P) - \sum_{S=2}^{\infty}(1 - Q_1)\left(\varphi_1(S)\right) \\ &= \log(3) - (1/3) \cdot \sum_{S=2}^{\infty}\left((2/3)^S - 2(1/2)^S\right)\log(S) \\ &\approx 1{,}58496 - 0{,}308518 \approx 1.27644\, bit. \end{aligned}$$

A partir do Teorema 3.4 é possível definir

$$\begin{aligned} H_\iota(K_{m_1,m_2,\ldots,m_k}, P) &= H(P) - \sum_{S=2}^{\infty}\sum_{i:m_i\geq 2}(1 - Q_i)\varphi_i(S) \\ &= H(P) - L(P). \end{aligned}$$

Dessa maneira, há, pelo menos, $2^{nL(P)}$ sequências típicas de comprimento n que passam a pertencer a uma classe de equivalência devido às relações de dependência/independência. Quando o grafo G é vazio, $L(P) = 0$ de forma que cada palavra só gera ela mesma como classe. Quando o grafo G não é vazio, há um menor número de classes de intercâmbio, possibilitando uma compressão maior do que a definida pela entropia de Shannon que não considera símbolos com ordem parcial. Quando o grafo G completo, como foi visto, a entropia de intercâmbio é zero.

Corolário 3.1. *Assumindo uma fonte discreta e sem memória com distribuição de probabilidade sobre o conjunto de vértices* $V(G)$*. Se* G *não é um grafo vazio em* $V(G)$*, então* $H_\iota(G, P) < H(P)$*.*

Savari [25] apresenta desigualdades e o Teorema 3.4 que permite o cálculo exato da entropia de intercâmbio para o caso dos grafos $\overline{G}$ que são k-partidos completos. Entretanto, a entropia de intercâmbio é difícil de ser determinada para grafos mais gerais. Além disso, foi encontrada uma contradição no Teorema 3.3, quando aplicado diretamente. Parece haver algum problema ou falta de informações em seu enunciado. Segundo esse teorema e sabendo que $\tau_G(n) = |B_n|$, é esperado que

$$\limsup_{n\to\infty}\frac{\log\tau_G(n)}{n} \leq H_\iota(G, P). \tag{3.12}$$

Como exemplo, seja o polinômio de dependência do grafo G, complemento do grafo apresentado na Fig. 3.2 do Exemplo 3.2, dado por

$$D(G, z) = 1 - 3z + 1z^2$$

e a função geradora será

$$\frac{1}{D(G,z)} = \frac{1}{1-3z+1z^2} = \sum_{n=0}^{\infty} U_n\left(\frac{3}{2}\right) z^n,$$

em que $U_n(\cdot)$ é o polinômio de Chebyshev. Desse modo, o número de classes de intercâmbio de tamanho n será

$$\tau_G(n) = U_n\left(\frac{3}{2}\right)$$

e

$$\limsup_{n\to\infty} \frac{\log \tau_G(n)}{n} = \limsup_{n\to\infty} \frac{\log(U_n\left(\frac{3}{2}\right))}{n} = 1{,}3884\, bit.$$

Desta forma, o Teorema 3.3 não se verifica, pois

$$1{,}3884\, bit \geq 1{,}27644\, bit.$$

O mesmo foi observado quando a entropia foi calculada considerando outros grafos de não-comutatividade que preenchem os requisitos do Teorema 3.3.

3.3 Entropia de Comutatividade

O objetivo desta seção é apresentar uma definição de entropia para alfabetos com ordem parcial que supere os problemas observados nas definições anteriores. A principal condição para isso é que seja possível atribuir uma probabilidade às sequências e, por consequência, também às classes de equivalência de forma a permitir diferentes distribuições de probabilidade.

Para definir adequadamente uma medida de probabilidade às classes de equivalência, é necessário, primeiramente, definir uma fonte parcialmente comutativa.

Definição 3.5 (Fonte parcialmente comutativa)**.** *Seja um alfabeto finito* Σ *e um grafo de comutatividade* G*. Uma fonte parcialmente comutativa, representada por* $S(G, \Pr)$*, é aquela que pares de símbolos representados nas arestas de* G *comutam (i.e., se* $ab \equiv_G ba$*, então* ab *e* ba *representam a mesma mensagem). Além disso, os símbolos são emitidos de acordo com a distribuição* $\Pr$*.*

Desta forma, uma fonte parcialmente comutativa pode emitir até $\tau_G(n)$ mensagens distintas com comprimento n. Além disso, é possível definir a probabilidade de uma fonte parcialmente comutativa emitir uma mensagem $\mathbf{m}$. Como todas as sequências de uma mesma classe são equivalentes, então a probabilidade da mensagem será igual a soma das probabilidades.

Definição 3.6 (Probabilidade de mensagens)**.** *A probabilidade de uma fonte parcialmente comutativa* $S(G, \Pr)$ *sem memória emitir uma mensagem* $\mathbf{u}$ *corresponde ao envio de um elemento da classe* $\mathcal{E}_G(\mathbf{u})$*, assim*

$$\Pr(\mathcal{E}_G(\mathbf{u})) = \sum_{\mathbf{v}\in\mathcal{E}_G(\mathbf{u})} \Pr(\mathbf{v}) = |\mathcal{E}_G(\mathbf{u})| \cdot \Pr(\mathbf{u}). \tag{3.13}$$

Tendo definido a probabilidade de uma classe de equivalência, pode-se partir para a definição proposta para a entropia de comutatividade.

Definição 3.7 (Entropia de Comutatividade)**.** *Para uma fonte parcialmente comutativa* $S(G, \Pr)$*, a entropia é definida como:*

$$H(G, \Pr) = -\lim_{n\to\infty} \sum_{\mathcal{E}_G(\mathbf{u})\subset\mathcal{M}^n(\Sigma,G)} n^{-1} \Pr(\mathcal{E}_G(\mathbf{u})) \log\left(\Pr(\mathcal{E}_G(\mathbf{u}))\right) \tag{3.14}$$

Para uma fonte parcialmente comutativa sem memória, (3.14) pode ser reescrita como:

$$H(G, \Pr) = -\lim_{n\to\infty} \sum_{\mathbf{u}\in\mathcal{M}^n(\Sigma,G)} n^{-1} |\mathcal{E}_G(\mathbf{u})| \cdot \Pr(\mathbf{u}) \log\left(|\mathcal{E}_G(\mathbf{u})| \cdot \Pr(\mathbf{u})\right). \tag{3.15}$$

Nesta seção só serão consideradas fontes sem memória. Usando o fato de que para uma sequência $\mathbf{u}, |\mathbf{u}| = n$ do tipo Q (ver Apêndice C), sua probabilidade só depende do seu tipo e é dada por

$$\Pr(\mathbf{u}) = 2^{-nH(Q)}, \tag{3.16}$$

em que $H(Q)$ é a entropia de Shannon considerando o tipo de $\mathbf{u}$, ou seja, da frequência relativa dos símbolos de Σ presentes em $\mathbf{u}$. Então, tem-se

$$H(G, \Pr) = -\lim_{n\to\infty} \sum_{\mathbf{u}\in\mathcal{M}^n(\Sigma,G)} n^{-1} |\mathcal{E}_G(\mathbf{u})| \cdot 2^{-nH(Q)} \log\left(|\mathcal{E}_G(\mathbf{u})| \cdot 2^{-nH(Q)}\right) \tag{3.17}$$

$$= -\lim_{n\to\infty} \sum_{\mathbf{u}\in\mathcal{M}^n(\Sigma,G)} n^{-1} \frac{|\mathcal{E}_G(\mathbf{u})|}{2^{nH(Q)}} \left(\log|\mathcal{E}_G(\mathbf{u})| + \log 2^{-nH(Q)}\right) \tag{3.18}$$

$$= -\lim_{n\to\infty} \sum_{\mathbf{u}\in\mathcal{M}^n(\Sigma,G)} \frac{|\mathcal{E}_G(\mathbf{u})|}{n2^{nH(Q)}} \log|\mathcal{E}_G(\mathbf{u})| - \frac{|\mathcal{E}_G(\mathbf{u})|}{2^{nH(Q)}} H(Q) \tag{3.19}$$

Para um alfabeto totalmente comutativo, é esperado que a entropia de comutatividade seja igual a zero. Nesse caso, o número de elementos de cada classe será igual a $|\mathcal{E}_G(\mathbf{u})| = 2^{nH(Q)}$ e

$$H(G,\Pr) = -\lim_{n\to\infty} \sum_{\mathbf{u}\in\mathcal{M}^n(\Sigma,G)} \frac{|\mathcal{E}_G(\mathbf{u})|}{n2^{nH(Q)}} \log|\mathcal{E}_G(\mathbf{u})| - \frac{|\mathcal{E}_G(\mathbf{u})|}{2^{nH(Q)}} H(Q) \tag{3.20}$$

$$= -\lim_{n\to\infty} \sum_{\mathbf{u}\in\mathcal{M}^n(\Sigma,G)} \frac{2^{nH(Q)}}{n2^{nH(Q)}} \log 2^{nH(Q)} - \frac{2^{nH(Q)}}{2^{nH(Q)}} H(Q) \tag{3.21}$$

$$= -\lim_{n\to\infty} \sum_{\mathbf{u}\in\mathcal{M}^n(\Sigma,G)} \frac{1}{n} \log 2^{nH(Q)} - H(Q) \tag{3.22}$$

$$= -\lim_{n\to\infty} \sum_{\mathbf{u}\in\mathcal{M}^n(\Sigma,G)} \frac{1}{n} nH(Q) - H(Q) = 0. \tag{3.23}$$

Para um alfabeto totalmente não-comutativo, é esperado que a entropia de comutatividade seja igual a entropia de Shannon. Nesse caso, o número de elementos de cada classe será igual a $|\mathcal{E}_G(\mathbf{u})| = 1$, pois não há permutações permitidas, e, considerando uma distribuição equiprovável, $\Pr(\sigma) = 1/|\Sigma|, \sigma \in \Sigma$, tem-se

$$H(G,\Pr) = -\lim_{n\to\infty} \sum_{\mathcal{E}_G(\mathbf{u})\subset\mathcal{M}^n(\Sigma,G)} n^{-1} \Pr(\mathcal{E}_G(\mathbf{u})) \log\left(\Pr(\mathcal{E}_G(\mathbf{u}))\right) \tag{3.24}$$

$$= -\lim_{n\to\infty} \sum_{\mathbf{u}\in\Sigma^n} n^{-1} |\mathcal{E}_G(\mathbf{u})| \cdot \Pr(\mathbf{u}) \log\left(|\mathcal{E}_G(\mathbf{u})| \cdot \Pr(\mathbf{u})\right) \tag{3.25}$$

$$= -\lim_{n\to\infty} \sum_{\mathbf{u}\in\Sigma^n} n^{-1} \Pr(\mathbf{u}) \log\left(\Pr(\mathbf{u})\right) \tag{3.26}$$

$$= -\lim_{n\to\infty} \sum_{\mathbf{u}\in\Sigma^n} n^{-1} \frac{1}{|\Sigma|^n} \log \frac{1}{|\Sigma|^n} \tag{3.27}$$

$$= -\lim_{n\to\infty} n^{-1} \log|\Sigma|^{-n} \tag{3.28}$$

$$= \log|\Sigma|. \tag{3.29}$$

Por fim, considerando que as classes de equivalência sejam equiprováveis, é esperado obter o limite superior para a entropia de comutatividade, que é igual a entropia topológica do monoide de comutatividade. Nesse caso, tem-se que $\Pr(\mathcal{E}_G(\mathbf{u})) = 1/\tau_G(n)$ e

$$H(G,\Pr) = -\lim_{n\to\infty} \sum_{\mathcal{E}_G(\mathbf{u})\subset\mathcal{M}^n(\Sigma,G)} n^{-1} \Pr(\mathcal{E}_G(\mathbf{u})) \log\left(\Pr(\mathcal{E}_G(\mathbf{u}))\right) \tag{3.30}$$

$$= -\lim_{n\to\infty} \sum_{\mathcal{E}_G(\mathbf{u})\subset\mathcal{M}^n(\Sigma,G)} n^{-1} \frac{1}{\tau_G(n)} \log\left(\frac{1}{\tau_G(n)}\right) \tag{3.31}$$

$$= -\lim_{n\to\infty} n^{-1} \log\left(\frac{1}{\tau_G(n)}\right) \tag{3.32}$$

$$= \lim_{n\to\infty} n^{-1} \log \tau_G(n), \tag{3.33}$$

que, por sua vez, é semelhante à definição da entropia topológica.

A definição de entropia apresentada parece possuir as propriedades esperadas para o contexto de comutação e ordem parcial. Mas, como é possível observar em (3.15), o cálculo da entropia de comutatividade passa a depender do número de elementos de cada uma das classes. Sabe-se que determinar o número de elementos de uma classe de comprimento n é um problema #P-completo (ver Teorema 2.3), o que já é um problema difícil. É necessário determinar a qual classe de complexidade pertence o problema de definir o número de elementos para classes de equivalência de comprimento n. Intuitivamente, deve ser tão difícil quanto os problemas da classe #P-completo.

Há a alternativa de estimar um limite superior para o número de elementos de uma classe, como no Teorema 2.5. Isso poderia ser utilizado em uma aproximação do valor da entropia de comutatividade, mas esse teorema garante os limitantes para uma sequência individual. No máximo, seria possível gerar as classes de um comprimento grande e aproximar o valor da entropia. Talvez esse procedimento seja útil em algumas aplicações.

3.4 Conclusões

Neste capítulo foi realizada uma revisão bibliográfica acerca das entropias destinadas aos alfabetos com ordem parcial. Foi observado que as definições anteriores de entropia para esses tipos de alfabetos possuem limitações. A primeira analisada foi a entropia topológica que não assume distribuições de probabilidade. Foi observado que a entropia topológica do monoide corresponde ao limite superior para uma entropia de comutatividade. A segunda foi a entropia de traços infinitos que é baseada em suposições que limitam os casos de aplicação. Já a entropia de intercâmbio, possui alguns problemas de ordem prática, envolvendo as dificuldades de calcular a complexidade de Kolmogorov que é não-computável e algumas propriedades que contradizem o limite superior esperado para a entropia.

Foi proposta uma nova definição de entropia envolvendo alfabetos com ordem parcial, a entropia de comutatividade. Essa nova definição apresenta as propriedades esperadas para uma entropia envolvendo alfabetos com ordem parcial, de forma que a entropia de Shannon e a entropia topológica fazem parte de casos particulares da definição. Outra vantagem da definição proposta é que ela permite o cálculo numérico, apesar de ser geralmente difícil calcular já que envolve a contagem do número de elementos de uma classe que é um problema #P-completo.

Capítulo 4

Algoritmos Para Compressão de Sequências Parcialmente Comutativas

Neste capítulo serão apresentadas as principais técnicas voltadas para a compressão de sequências definidas em alfabetos com ordem parcial. Será apresentado também o algoritmo de Lempel-Ziv (LZ78) e suas propriedades. Em seguida, será apresentado o algoritmo que generaliza o LZ78 para alfabetos com ordem parcial, chamado de Lempel-Ziv com ordem parcial (POLZ). Serão determinadas as principais desigualdades relacionadas ao algoritmo proposto, bem como exemplos de utilização. Essas desigualdades exploram os limitantes obtidos no Capítulo 2.

4.1 Introdução

Os métodos mais populares de compressão são baseados em alfabetos com ordem total [28, 29, 30, 31]. O mesmo pode ser observado em técnicas mais atuais de compressão [31, 32, 33, 34, 35, 36, 37, 38, 39]. Essas abordagens são mais apropriadas para comprimir sequências associadas aos sistemas sequenciais. Entretanto, é notável o aumento do número de sistemas concorrentes (*e.g.,* computadores, celulares, redes de computadores, GPU). Em sistemas nos quais há ordem parcial entre eventos, como nos concorrentes, técnicas de compressão que levam em consideração essa característica são mais apropriadas, podendo levar a maiores taxas de compressão [25].

O problema de como realizar a compressão das palavras definidas em um alfabeto parcialmente comutativo foi apresentado primeiramente por Alur et al. [27]. Neste contexto, esses autores propuseram quatro algoritmos para compressão sem perdas que consideram esses alfabetos. O primeiro algoritmo usa o SEQUITUR como inspiração e é um algoritmo guloso. O segundo é um algoritmo *off-line* que substitui os pares mais frequentes em uma sequência e cria uma gramática usando os pares como regras. Os pares podem ser dependentes ou independentes. Os dois últimos algoritmos utilizam o conceito de projeção para

comprimir as sequências. Eles obtiveram resultados experimentais que indicam melhorias na compressão. Apesar de fornecer resultados melhores que os algoritmos de compressão de ordem total, os algoritmos de Alur et al. [27] não são ótimos. Desta forma, não é garantido que alcancem a máxima taxa de compressão, principalmente o que se baseia no SEQUITUR, um algoritmo sub-ótimo nesse sentido.

Savari [25] propôs utilizar a forma normal de Foata (FNF) [5] seguida de um método de compressão sem perdas. Na FNF, os símbolos que comutam são agrupados em blocos chamados de fatores de Foata. Esse agrupamento faz surgir padrões que favorecem o processo de compressão. A ideia descrita por Savari é mapear uma sequência em sua forma normal e substituir os fatores de Foata por elementos de um alfabeto, formando um dicionário. Com isso, espera-se obter uma representação mais compacta da sequência original. Entretanto, Savari não forneceu mais detalhes acerca da implementação e nem da eficiência desse esquema de compressão.

Reznik [40, 41] propôs um método de codificação de conjuntos de palavras não-ordenados. Em outras palavras, ele abordou o problema da decodificação de palavras sem necessariamente estar na mesma ordem em que apareceram na sequência não codificada. Ele usou uma representação em árvore e uma árvore de busca digital (DST) para comprimir o conjunto de palavras, mostrando que a abordagem requer $\log(m!)$ menos bits do que uma abordagem de ordem total, na qual m é a cardinalidade do conjunto de palavras. A principal limitação dos trabalhos de Reznik [40, 41] é que se aplicam somente quando todos os símbolos podem comutar, sendo um caso particular da ordem parcial, assim como a ordem total também o é.

Considerando os algoritmos sequenciais de compressão, a família de algoritmos de Lempel-Ziv (LZ) são baseados nos trabalhos de Lempel e Ziv [42, 28, 29]. Um desses algoritmos é o apresentado por Lempel e Ziv em 1978 [29], também conhecido como LZ78. Esse algoritmo foi utilizado como base para desenvolver um método de compressão para alfabetos com ordem parcial. Desta forma, a próxima seção descreve o LZ78 e algumas das suas propriedades.

4.2 Algoritmo de Lempel-Ziv

O LZ78 é um algoritmo de compressão sem perdas que, deterministicamente, gera um dicionário a partir de uma sequência de entrada [29]. A compressão é obtida por meio da decomposição dessa sequência em frases (subsequências) e codificando-as. As frases são as menores sub-sequências ainda não observadas ao percorrer a sequência da esquerda para a direita.

Seja Σ um alfabeto com ordem total. Uma sequência $S \in \Sigma^n$ é decomposta em m frases, tais como $S = S_1,S_2,\ldots,S_m$. O dicionário é formado por m tuplas da forma

(ρ, σ) e cada uma delas codifica uma frase $S_i, i \in [1, m]$. A codificação de uma frase S_i é obtida sabendo que S_i é composta de uma frase S_ρ observada anteriormente (i.e., $i > \rho$) concatenada com um elemento $\sigma \in \Sigma$, ou ainda $S_i = S_\rho\sigma$. A complexidade de uma sequência S é então definida como o número de frases $C(S) = m$ em que ela é decomposta.

Considere a sequência $S = abbabbbaababaa$ como um exemplo. Assim, S será decomposta em $S = a,b,ba,bb,ab,bba,aba,baa$, tendo por complexidade $C(S) = 8$. O dicionário obtido de S é formado pelas tuplas apresentadas na Tabela 4.1. Para poder definir tuplas como $(0, \sigma)$ e facilitar a decodificação dos dicionários, define-se $S_0 = \epsilon$, a palavra vazia como a primeira frase observada. Desta forma, a tupla $(0, a)$ codifica a frase $S_1 = S_0a = \epsilon a = a$.

i	0	1	2	3	4	5	6	7	8
S_i	ϵ	a	b	ba	bb	ab	bba	aba	baa
(ρ, σ)		$(0,\ a)$	$(0,\ b)$	$(2,\ a)$	$(2,\ b)$	$(1,\ b)$	$(4,\ a)$	$(5,\ a)$	$(3,\ a)$

Tabela 4.1 – Dicionário obtido para a sequência $S = abbabbbaababaa$.

Lempel e Ziv [42] encontraram um limite superior para o valor de complexidade de qualquer sequência de comprimento n definido em um alfabeto Σ. Este limite superior é indicado em Teorema 4.1.

Teorema 4.1. *Limite Superior para a complexidade de LZ78 [42, p.77] Para todas as sequências* $S \in \Sigma^n, \alpha = |\Sigma|$,

$$C(S) \leq \frac{n}{(1 - \epsilon_n) \log_\alpha n} \tag{4.1}$$

em que

$$\epsilon_n = 2\frac{1 + \log_\alpha \log_\alpha(\alpha n)}{\log_\alpha(n)}. \tag{4.2}$$

No Teorema 4.2 é demonstrado que o LZ78 é assintoticamente ótimo. Em outras palavras, a taxa de compressão do LZ78 se aproxima da taxa de entropia da fonte à medida que o comprimento da sequência aumenta.

Teorema 4.2. *[43, p.455] Seja* $\{\sigma_i\}_{-\infty}^{\infty}$ *um processo estocástico estacionário e ergódico. Seja* $l(\sigma_0, \sigma_1, \ldots, \sigma_{n-1})$ *o comprimento de uma palavra-código do LZ78 associada com a sequência* $\sigma_0, \sigma_1, \ldots, \sigma_{n-1}$. *Então,*

$$\limsup_{n\to\infty} \frac{1}{n} l(\sigma_0, \sigma_1, \ldots, \sigma_{n-1}) \leq H(\Sigma) \tag{4.3}$$

com probabilidade 1, em que $H(\Sigma) = \lim_{n\to\infty} n^{-1} H(\sigma_0, \sigma_1, \ldots, \sigma_{n-1})$ *é a taxa de entropia do processo.*

A taxa $H(\Sigma)$ pode ser interpretada como uma medida da quantidade de informação média obtida quando há uma nova frase na sequência.

4.3 Lempel-Ziv com ordem Parcial

O algoritmo proposto consiste em dividir uma sequência de entrada S em frases, $S = S_1, S_2, \cdots, S_m$, passando apenas uma vez em S. Cada frase $S_i, m \geq i \geq 1$ é a menor frase observada até agora na sequência que é não-coerente às frases anteriores, $S_j, i > j \geq 1$, de acordo com um grafo de comutatividade G. Estas frases são codificadas e armazenadas em um dicionário.

A cada nova frase observada na sequência de entrada S, é necessário verificar se ela é congruente com qualquer frase presente no dicionário. Para verificar a congruência entre duas frases, é necessário comparar suas projeções nos vértices do grafo de não-comutatividade e o tipo de frases, como indicado no Teorema 2.1.

Para verificar se duas frases são do mesmo tipo, pode-se usar um algoritmo de ordenação lexicográfica. Isso porque se duas frases forem do mesmo tipo, então serão iguais uma à outra quando ordenadas. Consequentemente, pode-se usar um algoritmo como o *quick-sort* que possui uma complexidade temporal de $O(m_i \log m_i)$, onde $|S_i| = m_i$ é o comprimento da frase a ser ordenada. Realizar a projeção de uma frase S_i de tamanho m_i requer uma complexidade temporal de $O(m_i|\overline{E}|)$, em que $|\overline{E}|$ é o número de arestas do grafo de não-comutatividade $\overline{G}$.

O algoritmo proposto gera um dicionário que consiste de m tuplas da forma (ρ_i, σ_i). Cada tupla codifica uma frase $S_i = \sigma_0 \ldots \sigma_{(m_i-1)}, i \in [1, m]$ de S, sendo o comprimento de S_i igual a $|S_i| = m_i$. As tuplas (ρ_i, σ_i) são tais que $S'_i = \sigma_0 \ldots \sigma_{(m_i-2)}$ é congruente a uma frase observada anteriormente S_{ρ_i}, ou seja $S'_i \equiv_G S_{\rho_i}$. Além disso, $\sigma_i = \sigma_{(m_i-1)}$.

Como exemplo, seja um grafo de comutatividade $G = (V, E)$ em que $V = \{a, b, c\}$ e $ab \equiv_G ba$. Agora supondo uma sequência de entrada $S = abaacbaabc$. O dicionário obtido observando a sub-sequência $abaacba$ é apresentado na Tabela 4.2. Para poder definir tuplas como $(0, \sigma_i)$ e facilitar a decodificação dos dicionários, assim como no dicionário gerado pelo LZ78, define-se $S_0 = \epsilon$, a palavra vazia como a primeira frase observada. Desta forma, a tupla $(0, a)$ codifica a frase $S_1 = S_0 a = \epsilon a = a$.

i	0	1	2	3	4	5
S_i	ϵ	a	b	aa	c	ba
(ρ, σ)		$(0,\ a)$	$(0,\ b)$	$(1,\ a)$	$(0,\ c)$	$(2,\ a)$

Tabela 4.2 – Representação do dicionário gerado pelo algoritmo proposto observando a sequência de entrada até $abaacba$.

Após a geração do dicionário para $abaacba$, o processo continua considerando o restante da sequência, abc. A candidata a uma nova frase é a, que já está no dicionário. Desta forma, a próxima candidata é ab. Neste ponto, ab é congruente a $S_5 = ba$. A próxima candidata é, então, abc. Não há nenhuma frase congruente a abc no dicionário e, portanto,

$S_6 = abc$ é a nova menor frase observada até agora na sequência. Para codificar S_6, usa-se o fato de que $S'_6 = ab$ é congruente a S_5. Assim, pode-se concluir que $\rho_6 = 5$, $\sigma_6 = c$, e S_6 é codificada como $(5, c)$.

Levando em consideração a complexidade temporal da implementação, o algoritmo proposto utiliza um dicionário auxiliar que consiste em tuplas (t_i, π_i). O termo t_i corresponde ao tipo da frase e é obtida ordenando S_i. O termo π_i corresponde ao conjunto de projeções de S_i nas arestas de $\overline{G}$. Desta forma, duas tuplas (t_i, π_i) e (t_j, π_j) são iguais se e, somente se, $S_i \equiv_G S_j$. Esse dicionário auxiliar será utilizado para verificar se uma candidata a nova frase é congruente a uma frase já observada ou não.

Utilizar apenas um dicionário, como no LZ78, implicaria em realizar as projeções das frases presentes nesse dicionário toda vez que fosse necessário verificar se uma frase é nova ou não. Assim, quando uma frase é nova, o algoritmo armazena no dicionário auxiliar seu tipo e projeção. Isto lhe permite verificar em tempo linear se duas frases são congruentes.

Assim, a partir de uma sequência S, o algoritmo proposto gera um dicionário onde não há duas frases codificadas que são congruentes, de acordo com um grafo G. A complexidade $C(S, G)$ da sequência S de acordo com um grafo G é o número de frases não-congruentes no dicionário, de forma semelhante ao algoritmo LZ78.

Quando o grafo de não-comutatividade for completo, a fatoração em frases obtida pelo algoritmo proposto é igual àquela obtida usando o LZ78. Assim, o algoritmo proposto generaliza LZ78 para considerar sequências formadas por símbolos com uma ordem parcial. Portanto, o algoritmo proposto será chamado de Lempel-Ziv (POLZ) com Ordem Parcial.

O processo de decodificação do POLZ é semelhante ao LZ78, mas produz uma sequência *equivalente* S_{eq} ao invés da sequência original S, ou ainda, $S_{eq} \equiv_G S$. No procedimento a seguir, descreve-se como obter a sequência equivalente S_{eq}. Seja L uma lista chamada de lista de decodificação. A notação $L[i]$ representa i-ésimo elemento de L. Por definição, o primeiro elemento da lista L é a palavra vazia e os índices da lista começam a partir do zero. À vista disso, $L = [\epsilon]$ ou $L[0] = \epsilon$. Além disso, novos elementos sempre são adicionados no final de L. A partir das tuplas (ρ_i, σ_i) do dicionário que codifica as frases, reconstrói-se frases equivalentes a S_i por meio da concatenação $S_i = L[\rho_i]\,\sigma_i$, $i = 1, \ldots, C(G, S)$, e adiciona-se S_i em L. Após a obtenção de todas as frases S_i, a lista de decodificação será $L = \left[\epsilon, S_1, \ldots, S_{C(G,S)}\right]$. Como resultado, a sequência equivalente S_{eq} é formada por todas as frases de L concatenadas na mesma ordem que aparecem na lista.

Como um exemplo, considera-se novamente o grafo de comutatividade $G = (V, E)$ em que $V = \{a, b, c\}$ e $ab \equiv_G ba$. Para uma sequência de entrada igual a $S = abaacbaabc$, o dicionário codificado gerado pelo POLZ é apresentado na Tabela 4.3.

i	1	2	3	4	5	6
(ρ, σ)	$(0,\ a)$	$(0,\ b)$	$(1,\ a)$	$(0,\ c)$	$(2,\ a)$	$(5,\ c)$

Tabela 4.3 – Representação do dicionário gerado para a sequência *abaacbaabc*.

A primeira tupla é $(0,\ a)$, então $S_1 = L\,[0]\,a = \epsilon a = a$. Além disso, a é adicionado ao final da lista de decodificação L. A segunda tupla é $(0,\ b)$, então $S_2 = b$ e b é adicionado a L. A terceira tupla é $(1,\ a)$ e, então $S_3 = L\,[1]\,a$. Cabe notar que nesse ponto a lista de decodificação é $L = [\epsilon, a, b]$. Portanto, $L\,[1] = a$, $S_3 = aa$ e aa também é adicionado a L. Para as outras tuplas, o processo se repete, obtendo-se como resultado a lista L = [ϵ,*a*,*b*,*aa*,*c*,*ba*,*bac*]. Concatenando os elementos de L na mesma ordem, chega-se a $S_{eq} = abaacbabac$ e, como esperado, $S \equiv_G S_{eq}, abaacbaabc \equiv_G abaacbabac$.

4.3.1 Implementação do Algoritmo

O pseudocódigo da etapa de compressão do POLZ é apresentado no Algoritmo 2 e o pseudocódigo da etapa de descompressão é apresentado no Algoritmo 3. Para a etapa de compressão, é necessária a notação de sub-sequência. Dada uma sequência $S = \sigma_0\sigma_1 \ldots \sigma_{n-1}$, em que $\sigma_i \in \Sigma$ e $|S| = n$, uma sub-sequência de S é denotada como $S[i : j] = \sigma_i\sigma_{i+1} \ldots \sigma_{j-2}\sigma_{j-1}$, $0 \leq i \leq j \leq n-1$.

Para exemplificar como POLZ funciona, considera-se, como exemplo, a sequência S = *abbababababababaa*. Usando o LZ78, as frases serão

$$S = a,b,ba,bb,ab,bba,aba,baa \tag{4.4}$$

e codificado de acordo com as tuplas da Tabela 4.1, então a complexidade de S é $C(S) = 8$.

Supondo agora um grafo de não-comutatividade $\overline{G} = a - c - b$ ($ab \equiv_G ba$), então as frases de acordo com o POLZ serão

$$S = a,b,ba,bb,abb,baa,baba,a. \tag{4.5}$$

O dicionário obtido utilizando o POLZ com a sequência S = *abbababababababaa* e o grafo $\overline{G}$ é composto pelas tuplas apresentadas na Tabela 4.4.

Observa-se que até a quarta frase, as frases obtidas por ambos os algoritmos são iguais. Na quinta frase, entretanto, *ab* é congruente a *ba*. Portanto, a menor frase ainda não observada no caso do POLZ é *abb*. Desta forma, a complexidade de S usando a POLZ é igual a $C(G, S) = 7$. Cabe lembrar que, se o grafo de não-comutatividade fosse completo, as frases em ambos os algoritmos seriam iguais.

A partir do dicionário representado na Tabela 4.4, pode-se obter a lista de decodificação $L = [a, b, ba, bb, bab, baa, baba, a]$. Por consequência, a sequência decodificada será $S_{eq} = abbabbbabbaababaa$ e $S_{eq} \equiv_G S$.

Algoritmo 2: Pseudocódigo da compressão usando o POLZ.

Entrada: Sequência S, Grafo de não-comutatividade $\overline{G}$
Saída: Dicionário D, Complexidade $C(G,S)$ de S

$Q \leftarrow \emptyset$;
$D \leftarrow \emptyset$;
$\rho \leftarrow 0$;
$inc \leftarrow 1$;
$ind \leftarrow 0$;
$C(G,S) \leftarrow 0$;
enquanto *True* **faça**
  se $ind + inc > |S|$ **então**
    break;
  fim
  $S_i \leftarrow S[ind : ind + inc]$;
  $t_i \leftarrow S_i$ ordenado usando *quick-sort*;
  $\pi_i \leftarrow$ projeções de S_i nas arestas de $\overline{G}$;
  se $(t_i, \pi_i) \notin Q$ **então**
    $Q \leftarrow Q \cup (t_i, \pi_i)$;
    $\sigma_i \leftarrow$ último elemento de S_i;
    $D \leftarrow D \cup (\rho, \sigma_i)$;
    $C(G,S) \leftarrow C(G,S) + 1$;
    $ind \leftarrow ind + inc$;
    $inc \leftarrow 1$;
    $\rho \leftarrow 0$;
  senão
    $\rho \leftarrow \rho'$ tal que $(t_{\rho'}, \pi_{\rho'}) = (t_i, \pi_i)$ e $(t_{\rho'}, \pi_{\rho'}) \in Q$;
    $inc \leftarrow inc + 1$;
  fim
fim

Algoritmo 3: Pseudocódigo da descompressão usando o POLZ.

Entrada: Dicionário D
Saída: Sequência $S_{eq} \equiv_G S$

$S_{eq} \leftarrow \epsilon$;
$L \leftarrow [\epsilon]$;
$i \leftarrow 1$;
enquanto $i \leq |D|$ **faça**
  $S_{\rho_i} = L\,[\rho_i]$;
  $S_i \leftarrow S_{\rho_i}\sigma_i$;
  $L \leftarrow L \cup S_i$;
  $S_{eq} \leftarrow S_{eq}S_i$;
  $i \leftarrow i + 1$;
fim

i	0	1	2	3	4	5	6	7	8
S_i	ϵ	a	b	ba	bb	abb	baa	$baba$	a
(ρ, σ)		$(0,\ a)$	$(0,\ b)$	$(2,\ a)$	$(2,\ b)$	$(3,\ b)$	$(3,\ a)$	$(5,\ a)$	$(0,\ a)$

Tabela 4.4 – Representação do dicionário obtido na compressão usando POLZ.

4.3.2 Limitantes e Propriedades

Teorema 4.3. *Seja $\tau_G(n)$ o número de classes de equivalência de comprimento n relacionado ao grafo de comutatividade G e uma função $f(n) = \alpha\beta^n$ tal que*

$$\tau_G(n) \preccurlyeq f(n). \tag{4.6}$$

Essa notação é equivalente a $f(n) \geq \tau_G(n)$ para todo $n \geq n_0, n_0 \in \mathbb{N}^+$. Então, a complexidade de qualquer sequência $S \in \Sigma^n$ tem um limite superior dado por

$$C(G, S) \preccurlyeq \frac{n}{\log_\beta(n)\,(1 - \epsilon_n)}, \tag{4.7}$$

em que

$$\epsilon_n = \frac{\log_\beta(\alpha) + 3 + \log_\beta(\log_\beta(n))}{\log_\beta(n)}. \tag{4.8}$$

Demonstração. Seja $\tau_G(n)$ o número de classes equivalentes de comprimento n de acordo com o grafo de comutatividade $\overline{G}$ e $\tau_G(n) \preccurlyeq \alpha\beta^n$. Além disso, seja n_k a soma dos comprimentos de todas as frases não-congruentes de comprimento menor ou igual a k, então

$$\begin{aligned} n_k = \sum_{i=1}^{k} i\tau_G(i) &\preccurlyeq \sum_{i=1}^{k} i\alpha\beta^i \\ &= \frac{\alpha\beta}{\beta - 1}\left(\beta^k\left(k - \frac{1}{\beta - 1}\right) + \frac{1}{\beta - 1}\right). \end{aligned} \tag{4.9}$$

Consequentemente, uma sequência $S \in \Sigma^n$ de comprimento $n = n_k$ tem, no máximo,

$$N_k = \sum_{i=1}^{k} \tau_G(i) \preccurlyeq \sum_{i=1}^{k} \alpha\beta^i = \frac{\alpha\beta}{\beta - 1}(\beta^k - 1) \tag{4.10}$$

frases não-congruentes.

Pela definição de complexidade de uma sequência e considerando um grafo de comutatividade G, então $C(G,S) \leq N_k + 1$ e

$$\begin{aligned} C(G,S) \leq N_k + 1 &\preccurlyeq \frac{\alpha\beta}{\beta-1}(\beta^k - 1) + 1 \\ &< \frac{\alpha\beta}{\beta-1}\beta^k \\ &< \frac{n_k}{k - \frac{1}{\beta-1}} \leq \frac{n_k}{k-1}. \end{aligned} \tag{4.11}$$

Se $n_k \leq n \leq n_{k+1}$, pode-se expressar n como $n = n_k + \Delta_k$. A parcela Δ_k é menor que $(k+1)\alpha\beta^{k+1}$ e não causa o aumento do número de palavras não-congruentes por mais de $\Delta_k/(k+1)$, portanto

$$C(G,S) \preccurlyeq \frac{n_k}{k-1} + \frac{\Delta_k}{k+1} < \frac{n_k + \Delta_k}{k-1} = \frac{n}{k-1}. \tag{4.12}$$

Sabendo que

$$n \geq n_k = \frac{\alpha\beta}{\beta-1}\left(\beta^k k - \frac{\beta^k + 1}{\beta - 1}\right) > \beta^k, \tag{4.13}$$

então,

$$k < \log_\beta(n). \tag{4.14}$$

Além disso, tem-se que

$$n \leq n_{k+1} = \frac{\alpha\beta^{k+2}(k+1)}{\beta-1} - \frac{\beta^{k+1}+1}{(\beta-1)^2} < \alpha\beta^{k+2}(k+1), \tag{4.15}$$

portanto,

$$k < \log_\beta(n) < \log_\beta(n_{k+1}) < \log_\beta(\alpha) + k + 2 + \log_\beta(k+1). \tag{4.16}$$

Desta forma, obtém-se que

$$\begin{aligned} k - 1 &> \log_\beta(n) - \log_\beta(\alpha) - 3 - \log_\beta(k+1) \\ &> \log_\beta(n) - \log_\beta(\alpha) - 3 - \log_\beta(1 + \log_\beta(n)) \\ &> \log_\beta(n) - \log_\beta(\alpha) - 3 - \log_\beta(\log_\beta(n)) \\ &= \log_\beta(n)\left(1 - \left(\frac{\log_\beta(\alpha) + 3 + \log_\beta(\log_\beta(n))}{\log_\beta(n)}\right)\right) \\ &= \log_\beta(n)\left(1 - \epsilon_n\right). \end{aligned} \tag{4.17}$$

Substituindo (4.17) em (4.12), tem-se como resultado

$$C(G,S) \preccurlyeq \frac{n}{\log_\beta(n)\,(1-\epsilon_n)}, \tag{4.18}$$

em que $\epsilon_n = \frac{\log_\beta(\alpha)+3+\log_\beta(\log_\beta(n))}{\log_\beta(n)}$. □

Teorema 4.4. *Dada uma sequência $S \in \Sigma^n$ e um grafo de comutatividade G, então*

$$C(G,S) \leq C(S), \tag{4.19}$$

e a igualdade é obtida quando G é um grafo vazio ($|E| = 0$).

Demonstração. A soma dos comprimentos de todas as sequências distintas de comprimento menor ou igual a k é $n_k = \sum_{i=1}^{k} iq^i$. O maior número $C(S)$ de frases distintas considerando estas sequências é dado por $C(S) \leq \sum_{i=1}^{k} q^i$.

Agora considera-se um alfabeto q-ário parcialmente comutativo associado a um grafo de comutatividade G. Neste caso, a soma dos comprimentos de todas as sequências distintas que têm comprimento menor ou igual a k é dada por $n'_k = \sum_{i=1}^{k} i\tau_G(i) \leq n_k$.

Sabe-se que $\tau_G(i) \leq q^i$ e que a igualdade é alcançada quando $G = (V, E)$ é um grafo vazio, *i.e.*, $|E| = 0$. Desta forma, o número máximo de frases não-congruentes torna-se $C(G,S) \leq \sum_{i=1}^{k} \tau_G(i) \leq C(S)$. □

Corolário 4.1. *Se um grafo de comutatividade G é um grafo não-vazio, então*

$$C(G,S) < C(S), \tag{4.20}$$

Demonstração. Segue imediatamente de Teorema 4.4. □

Teorema 4.5. *O POLZ é assintoticamente ótimo.*

Demonstração. Para um processo estacionário e ergódico $\{\sigma_i\}_{-\infty}^{\infty}$, pelo Teorema 4.2, sabe-se

$$\limsup_{n\to\infty} \frac{1}{n} l(\sigma_0, \sigma_1, \ldots, \sigma_{n-1}) \leq H(\Sigma) \tag{4.21}$$

com probabilidade 1 quando se trata de um alfabeto Σ com ordem total, onde $l(\sigma_0, \sigma_1, \ldots, \sigma_{n-1})$ é o comprimento da palavra código gerado pelo LZ78 associada à sequência $S = \sigma_0, \sigma_1, \ldots, \sigma_{n-1}$ e $H(\Sigma)$ é a taxa de entropia do processo.

Como $l(x_0, x_1, \ldots, x_{n-1}) = n^{-1}C(S)(\log C(S) + 1)$, então pode-se escrever

$$\limsup_{n\to\infty} \frac{C(S)(\log C(S)+1)}{n} \leq H(\Sigma), \tag{4.22}$$

como apresentado em [43].

Considerando agora um grafo de comutatividade G, ao utilizar Teorema 4.4, obtém-se

$$\limsup_{n\to\infty} \frac{C(G,S)(\log C(G,S)+1)}{n} \leq H(\Sigma), \tag{4.23}$$

o que prova o teorema. □

Nota-se que para um grafo de comutatividade vazio $G = (V, E)$, (4.22) se torna igual a (4.23). Consequentemente, a taxa de compressão da POLZ é tão boa quanto a da LZ78 e melhor conforme o nível de concorrência aumenta.

4.3.3 Aplicações

Para exemplificar algumas das possíveis aplicações do POLZ, são apresentados quatro exemplos envolvendo ordem parcial, onde o POLZ sempre oferece maior compressão do que a LZ78. O primeiro exemplo diz respeito à compressão de uma sequência de comandos dada a um robô. Neste caso, todos os símbolos dos comandos comutam entre si. No segundo exemplo, considera-se o robô do primeiro exemplo, com algumas restrições na comutatividade entre os comandos. No terceiro exemplo, utiliza-se o POLZ para comprimir sequências de operações realizadas em caixas eletrônicos que operam independentemente. No último exemplo, é modelado um sistema em tempo real (RTS) que apresenta uma dependência entre tarefas. Para aplicar o POLZ, obtém-se o grafo de comutatividade do sistema, sendo possível aplicar o POLZ para comprimir as sequências de tarefas realizadas pelo sistema.

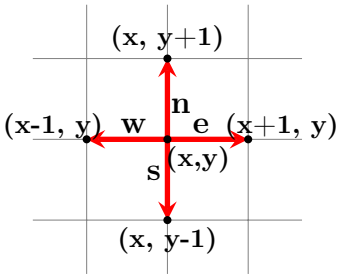
Comandos para Robôs

Um robô move-se em um plano cartesiano bidimensional e recebe uma sequência de comandos que definem sua trajetória de um ponto A para um ponto B. Sendo seu ponto inicial (x, y), os comandos podem ser

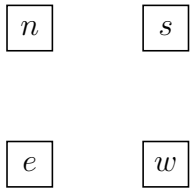
- n: incrementa em um sua coordenada no eixo y;
- s: decrementa em um sua coordenada no eixo y;
- e: incrementa em um sua coordenada no eixo x;
- w: decrementa em um sua coordenada no eixo x.

A representação gráfica dos comandos é apresentada na Fig. 4.1a. Após alguma experimentação, é possível observar que a ordem dos comandos em uma determinada sequência

não modifica o comportamento do robô ir do ponto A ao ponto B. Em termos de comutatividade, pode-se representar as relações como $nw \equiv_{G_1} wn$, $en \equiv_{G_1} ne$, $ne \equiv_{G_1} en$, $se \equiv_{G_1} es$, $ns \equiv_{G_1} sn$, $we \equiv_{G_1} ew$. Consequentemente, o grafo de não-comutatividade será vazio, como apresentado na Fig. 4.1b.



(a) Representação dos comandos.



(b) $\overline{G}_1$.

Figura 4.1 – Grafo de não-comutatividade dos comandos.

Como um exemplo, seja $S = nseww$ uma sequência de comandos. Como $\overline{G}$ é um grafo vazio, todas as $\binom{5}{1,1,1,2} = 60$ permutações de S são congruentes. Todas essas sequências descrevem trajetórias partindo do ponto inicial $A = (0,0)$ e chegando ao ponto $B = (-1,0)$. Algumas dessas sequências são representadas graficamente na Fig. 4.2.

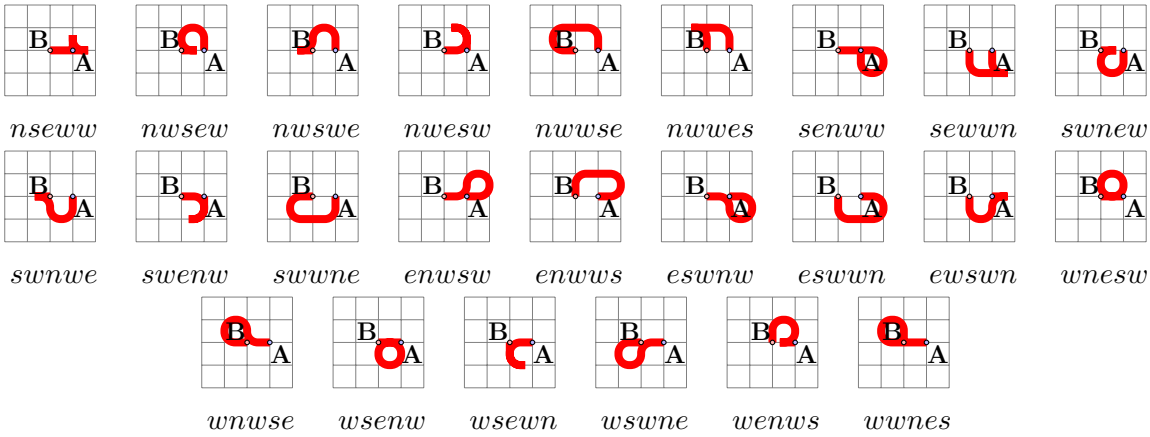


Figura 4.2 – Algumas sequências congruentes a $S = nseww$ de acordo com o grafo G_1.

A partir do grafo de comutatividade G_1, é possível determinar o polinômio de dependência de acordo com a Definição 2.4. Para G_1, que é um grafo completo, tem-se que $c_0 = 1$, $c_1 = 4$ (número de vértices), $c_2 = 6$ (número de arestas), $c_3 = 4$ (número de triângulos) e $c_4 = 1$. Desta forma, o polinômio de dependência é $D(G_1, z) = 1 - 4z + 6z^2 - 4z^3 + 1z^4$ e a função geradora do monoide será

$$\begin{aligned} \frac{1}{D(G_1,z)} &= \frac{1}{1-4z+6z^2-4z^3+1z^4} \\ &= \frac{1}{(z-1)^4} = \sum_{n=0}^{\infty} \frac{1}{6}(n+1)(n+2)(n+3)z^n. \end{aligned} \tag{4.24}$$

Na expressão (4.24), $\frac{1}{6}(n+1)(n+2)(n+3)$ representa o número exato de classes de comprimento n. Como a multiplicidade da menor raiz é 4, então, usando (2.17) para determinar uma aproximação assintótica, obtém-se

$$\tau_{G_1}(n) \sim \frac{1(-1)^4}{(4-1)!} n^{(4-1)} 1^{n+4} \tag{4.25}$$

$$= \frac{1}{6} n^3. \tag{4.26}$$

Desta forma, o número de classes de equivalência de comprimento n pode ser aproximado por $\tau_{G_1}(n) \sim \frac{1}{6}n^3$. Para poder utilizar o Teorema 4.3, a fim de determinar os limites para as complexidades $C(G_1, S)$, é necessário encontrar uma função $f(n) = \alpha\beta^n$ tal que $\tau_G(n) \preccurlyeq f(n)$. Qualquer função $f(n) = \beta^n$, com $\beta > 1$ preenche as especificações, mas o n_0, tal que $\tau_G(n_0) \leq f(n_0)$, é maior conforme β se aproxima de 1. Outro fator que ajuda a escolher $f(n)$ é considerar que, para o caso com ordem total, o número de classes seria 4^n. À vista disso, pode-se escolher um $f(n)$ que possui um $\beta \leq 4$ e já se obtém um limitante mais justo do que a complexidade do LZ78.

Para avaliar o desempenho do POLZ, foram geradas sequências aleatórias de comandos de comprimento l e suas complexidades foram verificadas. Levando em consideração as sequências de eventos geradas nesse exemplo, a função escolhida foi $f(n) = 1 \cdot 2^n$, de forma que $f(n) \geq \tau_{G_1}(n)$, para $n \geq 7$. As complexidades obtidas para o POLZ e LZ78 são apresentadas no gráfico da Fig. 4.3. Na mesma figura também são apresentados os limitantes superiores das complexidades para os dois algoritmos. Para a geração das sequências, os comandos foram tratados como variáveis independentes e identicamente distribuídas (iid).

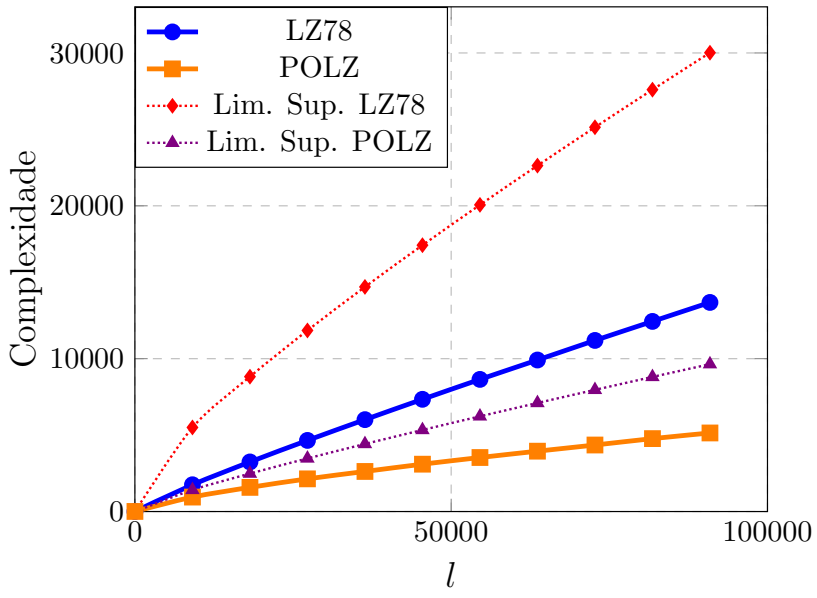


Figura 4.3 – Complexidade de sequências aleatórias de comandos.

Pode-se observar na Fig. 4.3 que a utilização do POLZ fornece uma complexidade menor que a metade da complexidade obtida usando o LZ78 quando as sequências passam do comprimento 15000. Portanto, a utilização do POLZ para comprimir sequências dos comandos leva a menores espaços de armazenamento de comandos quando comparado com o LZ78.

Comandos para Robôs com Restrição

Este exemplo utiliza o mesmo robô apresentado na Seção 4.3.3. Supondo agora que se queira evitar que este robô aja de certa forma, envolvendo a comutação de n a s e de w a e, gerando o grafo de não-comutatividade $\overline{G}_2$, representado na Fig. 4.4.

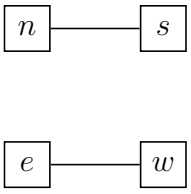


Figura 4.4 – Grafo de não-comutatividade $\overline{G}_2$.

Se a sequência de comandos é $S = snwwe$, então sequências equivalentes a S devem ter os mesmos tipos e as mesmas projeções que S nas arestas de $\overline{G}_2$. As projeções de S nas arestas de $\overline{G}_2$ são $\pi_{\{n,s\}}(S) = sn$ e $\pi_{\{w,e\}}(S) = wwe$. Desta forma, em todas as sequências equivalentes a S, s sempre ocorrerá antes de n e dois w ocorrerão antes de e. Consequentemente, as sequências equivalentes a S de acordo com G_2 são $snwwe$, $swnwe$, $swwne$, $swwnen$, $wsnwe$, $wswne$, $wswen$, $wwsne$, $wwsen$, $wwesn$, representadas graficamente na Fig. 4.5.

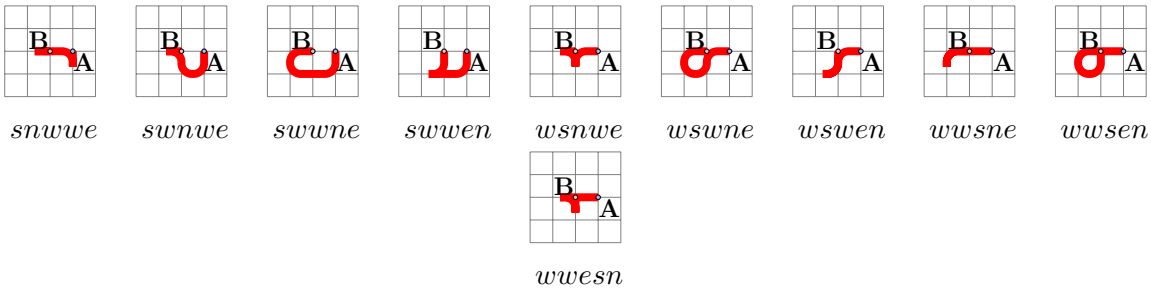


Figura 4.5 – Todas as sequências congruentes a $S = snwwe$ de acordo com $\overline{G}_2$.

O polinômio de dependência do grafo G_2 é $D(G, z) = 1 - 4z + 4z^2$ a função geradora do monoide será

$$\frac{1}{D(G_2, z)} = \frac{1}{1 - 4z + 4z^2} = \frac{(1/4)}{(z - 0.5)^2}. \tag{4.27}$$

A menor raiz real de $D(G_2, z)$ tem multiplicidade 2, sendo similar ao caso de (2.11), sendo possível obter

$$\frac{1}{D(G_2,z)} = \sum_{n=0}^{\infty}\left(\frac{1}{4}\right)\binom{n+1}{n}2^{n+2}z^n$$
$$= \sum_{n=0}^{\infty}(n+1)2^n z^n. \tag{4.28}$$

A partir de 4.28, conclui-se que o número de classes de equivalência é $\tau_{G_2}(n) = (n+1)2^n$. Para aplicar o Teorema 4.3, novamente é necessário determinar uma função $f(n)$ tal que

$$(n+1)2^n \preccurlyeq f(n), \tag{4.29}$$

a fim de obter os limitantes superiores para as complexidades $C(G_2, S)$ das sequências S.

Nesse caso, foi selecionado $f(n) = 2.1^n$, que é maior que $\tau_{G_2}(n)$ para $n > n_0 = 94$, satisfazendo (4.29). Como $\tau_{G_2}(n) \preccurlyeq f(n)$ e $f(n) = \alpha\beta^n = 1 \cdot 2.1^n$, agora pode-se aplicar o Teorema 4.3.

Sequências aleatórias de comandos foram geradas sob as mesmas condições que a seção 4.3.3 para calcular as complexidades dessas sequências. As complexidades obtidas e seus limitantes são apresentados na Fig. 4.6.

Como G_2 tem dois pares de comandos que não comutam, espera-se que a complexidade das sequências para este caso de acordo com o algoritmo POLZ (Upper POLZ) seja maior que o caso sem restrições, representado por G_1, ou seja, $C(G_1, S) \leq C(G_2, S)$.

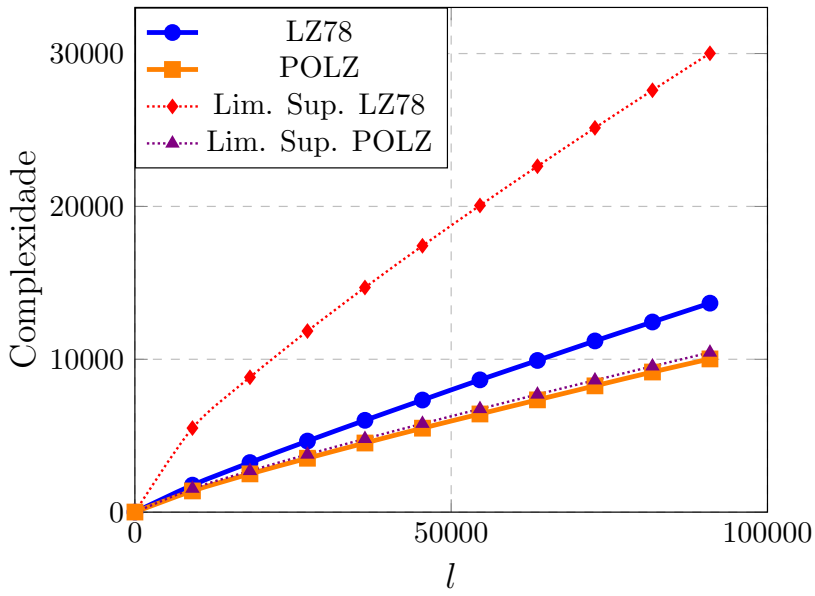


Figura 4.6 – Complexidade de sequências de comandos com restrição de comutatividade.

No entanto, $C(G_2, S)$ é menor do que a complexidade $C(S)$. Se o grafo G_2 fosse completo, então $C(G_2, S) = C(S)$. Os valores e limite superior para $C(S)$ são os mesmos do exemplo apresentado na Seção 4.3.3. O limite superior de acordo com o Teorema 4.1

é justo, devido à escolha de $f(n)$, mas é um limite válido para todo o comprimento l das sequências desde que a condição de (4.29) seja satisfeita.

Transações em Caixas Eletrônicos

Nesse exemplo, será necessário definir um modelo simples de caixa eletrônico (CE). Seja um conjunto de caixas eletrônicos (CE) que operam em diferentes locais da cidade. Em cada i-ésimo CE, o usuário pode realizar as operações depósito (D_i), saque (W_i) e checagem de saldo bancário (C_i). Além disso, o i-ésimo CE pode realizar internamente as operações de adicionar valor ao saldo bancário (A_i), subtrair um valor do saldo bancário (S_i), e solicitar saldo bancário (R_i) para o servidor.

Quando o usuário no i-ésimo CE deposita (D_i) uma quantia, o CE faz uma consulta (R_i) e depois adiciona (A_i) a quantia inserida pelo usuário ao saldo. Desta forma, a operação completa de depósito gera a sequência de comandos $D_iR_iA_i$. Ao verificar o saldo (C_i), o CE consulta o valor (R_i), então a operação completa gera a sequência de comando C_iR_i. Para fazer um saque (W_i), o CE faz uma consulta (R_i) e depois subtrai (S_i) o valor informado pelo usuário do saldo, tendo a sequência $W_iR_iS_i$. O comportamento do CE descrito pode ser representado pela máquina de estado finito (FSM) da Fig. 4.7.

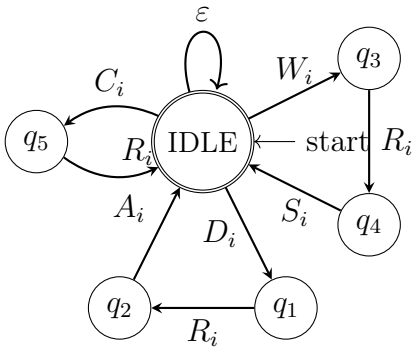


Figura 4.7 – Máquina de estados finitos representando um CE.

Nenhuma das operações realizadas no mesmo caixa eletrônico pode ter suas ordens trocadas. Como mostrado na Fig. 4.8, há uma dependência entre a ordem das operações de execução. Ignorar estas dependências leva a erros no saldo bancário do usuário. Assim, o grafo de não-comutatividade de cada caixa eletrônico tem as operações como vértices. Além disso, é um grafo completo, como apresentado na Fig. 4.8.

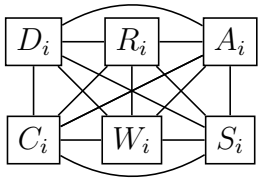


Figura 4.8 – Grafo de não-comutatividade $\overline{G}_i$ do i-ésimo CE.

O conjunto de CE possuem k caixas e cada CE tem um usuário ou não, realizando operações. Além disso, um CE realiza apenas uma operação de cada vez. Como exemplo, seja $k = 3$. Uma das sequências possíveis é $D_0W_1R_0D_2A_0R_1S_1R_2A_2$, representada na Tabela 4.5. A troca da posição das operações do mesmo caixa eletrônico produz resultados indesejados. A troca de posições entre operações de diferentes caixas eletrônicos produz o mesmo resultado final, levando a outras representações da execução de operações paralelas.

CE_i	Traço								
	D_0	W_1	R_0	D_2	A_0	R_1	S_1	R_2	A_2
0	D_0		R_0		A_0				
1		W_1				R_1	S_1		
2				D_2				R_2	A_2

Tabela 4.5 – Exemplo de sequência gerada considerando um conjunto com $k = 3$ CE.

Seja G_{CE} o grafo de comutatividade do sistema formado por k CE. Este grafo será um k-partido, em que cada partição é o grafo de comutatividade G_i relacionado ao i-ésimo CE. Consequentemente, G_{CE} é um grafo k-partido equilibrado com $k|\Sigma|$ vértices e o número de classes equivalentes é dado por

$$\tau_{G_{CE}}(n) = \binom{n+k-1}{n}|\Sigma|^n. \tag{4.30}$$

Assumindo que cada usuário no caixa eletrônico i-th, $i = 0, \ldots, m-1$, realiza as operações D_i, W_i e C_i com igual probabilidade. Além disso, enquanto as operações são realizadas em alguns CE, outros podem estar sem um usuário. O caso do CE estar sem um usuário é representado pela operação ϵ e é considerado uma operação completa. Assim, suponha que $\Pr(W_i) = Pr(D_i) = \Pr(C_i) = P(\epsilon) = 1/4$. Simulando a execução de operações completas em $k = 10$ CE em paralelo, foram obtidas sequências de operações de comprimento l. Os números considerados de operações completas em cada CE foram iguais a 10, 500, 1000, 2000, 3000, 4000, 5000, 6000, 7000, e 8000. As complexidades obtidas para essas sequências são apresentadas na Fig. 4.9.

Cabe observar que se $k = 1$, então os valores de complexidade obtidos usando POLZ e LZ78 seriam iguais. A partir do Corolário 4.1, segue que para qualquer $k > 1$, há uma redução na complexidade e ganhos na compressão quando se usa POLZ.

Sistema em Tempo Real

A análise da dinâmica de execução de programas concorrentes é difícil, por isso os registros (*logs*) de execução desses programas são utilizados para esse fim. Estes registros são grandes; consequentemente, é mais conveniente comprimir os arquivos para o armazenamento [44, 45].

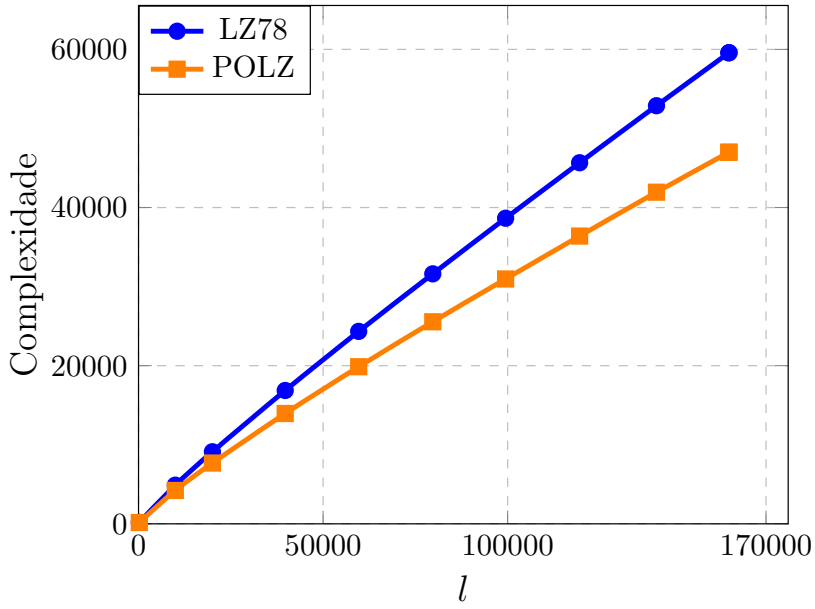


Figura 4.9 – Complexidade de sequência aleatórias de operações em 10 CE.

Como último exemplo, é considerado um sistema em tempo real modelado pelo grafo de precedência mostrado na Fig. 4.10. O grafo de precedência é um grafo orientado no qual cada vértice representa uma tarefa executada pelo sistema. Desta forma, se há uma aresta partindo do vértice T_i e chega em T_j, diz-se que a tarefa T_i precede T_j ou que T_j depende de T_i. O sistema representado na Fig. 4.10 consiste em 9 tarefas representadas por T_1 a T_9, formando o alfabeto $\Sigma = \{T_1, \ldots, T_9\}$. As tarefas T_1, T_5, T_7, T_9 não dependem de nenhuma outra tarefa.

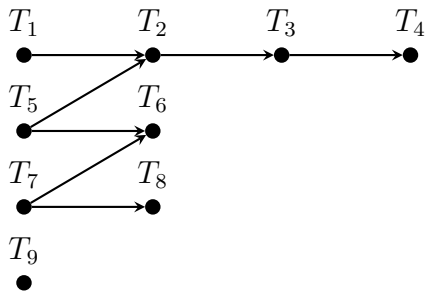


Figura 4.10 – Grafo de precedência do sistema.

Na Fig. 4.11 é mostrado como as tarefas do sistema podem ser executadas utilizando dois processadores independentes. Duas tarefas que não dependem uma da outra podem ser completadas em paralelo, enquanto que duas que têm dependência não podem. Este comportamento pode ser representado por uma relação comutativa parcial.

Para obter o grafo de não-comutatividade do sistema, pode-se observar que sequências que representam execuções das tarefas sempre conterão sub-sequências da forma uT_ivT_jw em que $T_i \to T_j$. Além disso, u, v, w são sub-sequências possivelmente vazias que representam execuções permitidas de tarefas. Como a ordem de T_i e T_j não pode ser trocada,

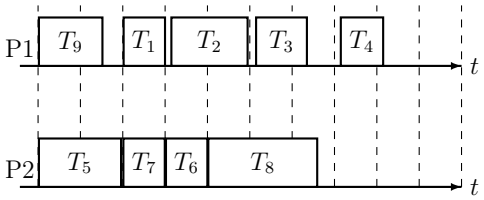


Figura 4.11 – Exemplo de execução das tarefas do sistema usando dois processadores.

então, existe dependência e, no grafo de não-comutatividade, T_i e T_j são conectados por uma aresta. De forma geral, para cada cadeia $T_i \to T_j \to \cdots \to T_l$ no grafo de precedência, todos os elementos da cadeia serão conectados por uma aresta no grafo de não-comutatividade, formando um clique. Analisando o grafo de precedência do sistema desse exemplo, as cadeias são $T_1 \to T_2 \to T_3 \to T_4$, $T_5 \to T_2 \to T_3 \to T_4$, $T_5 \to T_6$, $T_7 \to T_6$, $T_7 \to T_8$. A partir disso, é possível obter o grafo de não-comutatividade do sistema, apresentado na Fig. 4.12.

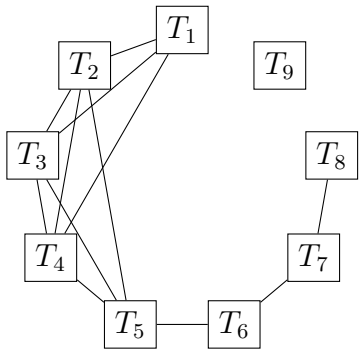


Figura 4.12 – Grafo de não-comutatividade $\overline{G}_3$.

A partir do grafo de não-comutatividade, pode-se usar POLZ e obter os limites para a complexidade das sequências, como nos casos anteriores.

4.4 Conclusões

Neste capítulo, foram apresentadas as principais abordagens destinadas à compressão de sequências definidas em alfabetos com ordem parcial. A partir dessa apresentação, foi possível concluir que técnicas disponíveis para esse fim não são ótimas ou são limitadas a casos particulares da ordem parcial. Desse modo, foi apresentado um novo algoritmo para compressão de sequências com ordem parcial, chamado de Lempel-Ziv com ordem parcial (POLZ). Também foram desenvolvidos limitantes para a complexidade de sequências usando esse novo algoritmo, sendo possível demonstrar que ele é ótimo. Por fim, foram apresentados exemplos de aplicações para evidenciar o ganho na compressão ao se considerar a ordem parcial presente na natureza dos sistemas considerados. No próximo capítulo serão apresentadas as conclusões obtidas nessa dissertação e serão discutidas as possibilidades de trabalhos futuros.

Capítulo 5

Conclusões

Na presente dissertação, foram apresentadas novas técnicas de enumeração e medida de informação que consideram alfabetos com ordem parcial. Essas técnicas visam aplicações em sistemas que podem ser modelados com esses alfabetos, principalmente sistemas concorrentes.

Foram apresentadas as definições necessárias a compreensão da dissertação. Dentre essas definições, foram apresentados temas como teoria da informação e complexidade de algoritmos. Também foram apresentados conceitos como monoides de comutatividade e pilha de peças que são formas de formalizar o tratamento de alfabetos parcialmente com ordem parcial.

Por meio da revisão bibliográfica, foi possível observar que existem poucas técnicas e algoritmos voltados para alfabetos com ordem parcial. Dentre as medidas de informação encontradas na bibliografia, ou elas são difíceis de serem determinadas ou se referem a casos particulares. Entretanto, essas técnicas mostram o potencial de considerar a ordem parcial ao invés de ordem total.

Como resultados, foi apresentado um método de estimação do número de classes de equivalência de um monoide de comutatividade, sendo possível obter uma expressão simples e assintoticamente equivalente à original. Esse resultado foi utilizado para obter limitantes para o número de classes de equivalência considerando uma família de grafos com $|V|$ vértices e $|E|$ arestas. Isso foi realizado como uma forma de contornar o problema de contar cliques que, por sua vez, são usados na estimação do número de classes de equivalência. Isso permite obter limitantes úteis em análises de algoritmos, por exemplo.

Foi proposta, também, uma nova definição para entropia de alfabetos com ordem parcial, chamada de entropia de comutatividade. Essa definição apresenta as características esperadas para uma entropia para tais alfabetos e possibilita superar as principais limitações das definições apresentadas anteriormente na literatura, como não ser somente aplicada a casos particulares e nem ser baseada em uma grandeza não computável. Foi

demonstrado que determinar o valor da entropia de comutatividade, envolve um problema de natureza pelo menos de complexidade #P-completo. Entretanto, é possível estimar o valor dessa entropia utilizando algoritmos de contagem do número de elementos de classes de equivalência.

Como último resultado, foi apresentado um novo algoritmo para a compressão de sequências definidas em um alfabeto com ordem parcial. Técnicas desenvolvidas, também nessa dissertação, foram aplicadas para provar que o algoritmo é ótimo. As aplicações do algoritmo em exemplos de sistemas demonstram o ganho na compressão ao considerar a natureza de ordem parcial dos processos.

Os resultados obtidos indicam que há ganhos em considerar a ordem parcial entre símbolos e medidas de informação podem ser úteis na análise de sistemas caracterizados por esse tipo de ordem. Medidas de informação que consideram ordem parcial terão o mesmo papel da entropia de Shannon na determinação de limitantes para a compressão e em problemas de contagem.

5.1 Trabalhos Futuros

A pesquisa sobre entropia de alfabetos parcialmente comutativos é uma área relativamente recente e existem tópicos que não foram totalmente exploradas e alguns que ainda não possuem solução. Algumas dessas questões em aberto são propostas como trabalhos futuros, das quais:

- **Determinar a classe de complexidade da determinação da entropia de comutatividade**

  Sabe que, dado uma palavra, determinar o número de elementos de sua classe é um problema #P-completo. Para determinar a entropia de comutatividade, é necessário saber o número de elementos de todas as classes de comprimento n. Saber a qual classe de complexidade esse problema pertence vai ajudar a direcionar abordagens de como calcular a entropia de comutatividade.

- **Obter uma estimação para o número de elementos de uma classe de equivalência de comprimento** n

  Foi visto que, dado um elemento, é possível estimar o número de elementos de sua classe. Essa estimação pode ser utilizada para aproximar a entropia de comutatividade, mas seria necessário gerar palavras de comprimentos grandes, tornando computacionalmente inviável. Uma alternativa para estimar a entropia de intercâmbio, então, é obter estimações para o número de elementos das classes de comprimento n.

- **Provar se a taxa de compressão do POLZ é igual a entropia de comutatividade**

  Nessa dissertação, foi demonstrado que a taxa de compressão obtida pelo POLZ é menor ou igual à taxa de entropia de Shannon. Entretanto, não foi demonstrado quantos bits de informação a menos são necessários para representar as sequências comprimidas. Seguindo a relação do LZ78 com a entropia de Shannon, é esperado que a taxa de compressão do POLZ se aproxime da entropia de comutatividade (para uma fonte ergódica), conforme os comprimentos das sequências de entrada aumentem.

5.2 Produção Científica

Como resultado deste trabalho, um artigo foi aceito e publicado em uma revista internacional e um foi publicado em um congresso nacional da área, sendo eles:

- "A New Algorithm for Compression of Partially Commutative Alphabets" [46]
  Autores: Andresso da Silva e Francisco M. de Assis
  Publicado em: Information Sciences, 2022.

- "Novos Algoritmos para a Compressão de Sequências Parcialmente Comutativas" [47]
  Autores: Andresso da Silva e Francisco M. de Assis

Aceito em: XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT), 2022.

Referências

1 SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, v. 27, p. 379–423, july, october 1948. Citado nas páginas 18 e 93.

2 MAZURKIEWICZ, A. Concurrent program schemes and their interpretations. *DAIMI Report Series*, v. 6, n. 78, Jul. 1977. Citado nas páginas 18, 21, 22 e 47.

3 MARINESCU, D. C. Chapter 4 - parallel and distributed systems. In: MARINESCU, D. C. (Ed.). *Cloud Computing (Second Edition)*. Second edition. Morgan Kaufmann, 2018. p. 113–150. ISBN 978-0-12-812810-7. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128128107000054>. Citado na página 19.

4 NAIK, N. Demystifying properties of distributed systems. In: *2021 IEEE International Symposium on Systems Engineering (ISSE)*. [S.l.: s.n.], 2021. p. 1–8. Citado na página 19.

5 PIERRE, C.; FOATA, D. *Problèmes combinatoires de commutation et réarrangements*. Berlin Heidelberg New York: Springer-Verlag, 1969. (Lecture notes in mathematics). ISBN 0-387-04604-6. Citado nas páginas 21, 25 e 56.

6 VIENNOT, G. X. Heaps of pieces, i: Basic definitions and combinatorial lemmas. *Annals of the New York Academy of Sciences*, v. 576, n. 1, p. 542–570, 1989. Disponível em: <https://nyaspubs.onlinelibrary.wiley.com/doi/abs/10.1111/j.1749-6632.1989.tb16436.x>. Citado nas páginas 22 e 28.

7 FISHER, D. C. The number of words of length n in a graph monoid. *Am. Math. Monthly*, Mathematical Association of America, USA, v. 96, n. 7, p. 610–614, ago. 1989. ISSN 0002-9890. Disponível em: <https://doi.org/10.2307/2325181>. Citado nas páginas 22 e 25.

8 FISHER, D. C.; SOLOW, A. E. Dependence polynomials. *Discrete Mathematics*, v. 82, n. 3, p. 251 – 258, 1990. ISSN 0012-365X. Disponível em: <http://www.sciencedirect.com/science/article/pii/0012365X9090202S>. Citado nas páginas 22, 25 e 26.

9 PERRIN, D. Words over a partially commutative alphabet. In: *Combinatorial Algorithms on Words. NATO ASI Series (Series F: Computer and Systems Sciences)*. [S.l.]: Springer, Berlin, Heidelberg, 1985. v. 12, p. 329–340. Citado na página 24.

10 GOLDWURM, M.; SANTINI, M. Clique polynomials have a unique root of smallest modulus. *Information Processing Letters*, v. 75, n. 3, p. 127–132, 2000. ISSN 0020-0190. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020019000000867>. Citado nas páginas 28 e 35.

11 BOUSQUET-MéLOU, M.; RECHNITZER, A. Lattice animals and heaps of dimers. *Discrete Mathematics*, v. 258, n. 1, p. 235–274, 2002. ISSN 0012-365X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0012365X02003527>. Citado na página 28.

12 VIENNOT, D. Emergent gravity and d-brane adiabatic dynamics: emergent lorentz connection. *Classical and Quantum Gravity*, IOP Publishing, v. 38, n. 24, p. 245004, nov 2021. Disponível em: <https://dx.doi.org/10.1088/1361-6382/ac337d>. Citado na página 28.

13 BERSTEL, J.; REUTENAUER, C. *Rational Series and Their Languages*. Berlin, Heidelberg: Springer-Verlag, 1988. ISBN 0387186263. Citado na página 33.

14 ERDőS, P. On the number of complete subgraphs contained in certain graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, p. 459–464, 1962. Citado na página 35.

15 FORD, D. R.; FULKERSON, D. R. *Flows in Networks*. USA: Princeton University Press, 2010. ISBN 0691146675. Citado nas páginas 39, 41 e 88.

16 FRANK, A. On chain and antichain families of a partially ordered set. *Journal of Combinatorial Theory, Series B*, v. 29, n. 2, p. 176–184, 1980. ISSN 0095-8956. Disponível em: <https://www.sciencedirect.com/science/article/pii/0095895680900799>. Citado nas páginas 39 e 88.

17 BOCHKOV, I. A.; PETROV, F. V. The bounds for the number of linear extensions via chain and antichain coverings. *Order*, v. 38, n. 2, p. 323–328, 2021. Disponível em: <https://doi.org/10.1007/s11083-020-09542-3>. Citado nas páginas 39 e 88.

18 MATTHEWS, P. Generating a random linear extension of a partial order. *The Annals of Probability*, Institute of Mathematical Statistics, v. 19, n. 3, p. 1367–1392, 1991. ISSN 00911798. Disponível em: <http://www.jstor.org/stable/2244487>. Citado na página 40.

19 RUSKEY, F. Generating linear extensions of posets by transpositions. *Journal of Combinatorial Theory, Series B*, v. 54, n. 1, p. 77–101, 1992. ISSN 0095-8956. Disponível em: <https://www.sciencedirect.com/science/article/pii/0095895692900678>. Citado na página 40.

20 PRUESSE, G.; RUSKEY, F. Generating linear extensions fast. *SIAM Journal on Computing*, v. 23, n. 2, p. 373–386, 1994. Disponível em: <https://doi.org/10.1137/S0097539791202647>. Citado na página 40.

21 BUBLEY, R.; DYER, M. Faster random generation of linear extensions. *Discrete Mathematics*, v. 201, n. 1, p. 81–88, 1999. ISSN 0012-365X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0012365X98003331>. Citado na página 40.

22 ONO, A.; NAKANO, S.-i. Constant time generation of linear extensions. In: LIŚKIEWICZ, M.; REISCHUK, R. (Ed.). *Fundamentals of Computation Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 445–453. ISBN 978-3-540-31873-6. Citado na página 40.

23 LUCA, A. de. On the entropy of a formal language. In: BARKHAGE, H. (Ed.). *Automata Theory and Formal Languages, 2nd GI Conference, Kaiserslautern, May 20-23, 1975*. Springer, 1975. (Lecture Notes in Computer Science, v. 33), p. 103–109. Disponível em: <https://doi.org/10.1007/3-540-07407-4_13>. Citado na página 45.

24 MANOHAR, R. The entropy of traces in parallel computation. *IEEE Transactions on Information Theory*, v. 45, n. 5, p. 1606–1608, 1999. Citado nas páginas 45 e 46.

25 SAVARI, S. Compression of words over a partially commutative alphabet. *IEEE Transactions on Information Theory*, v. 50, n. 7, p. 1425–1441, 2004. Citado nas páginas 47, 48, 49, 50, 55, 56 e 100.

26 SAVARI, S. A. On compressing interchange classes of events in a concurrent system. In: *Data Compression Conference, 2003. Proceedings. DCC 2003.* [S.l.: s.n.], 2003. p. 153–162. Citado na página 47.

27 ALUR, R. et al. Compression of partially ordered strings. In: *CONCUR 2003 - Concurrency Theory*. [S.l.]: Springer Berlin Heidelberg, 2003. p. 42–56. Citado nas páginas 47, 55 e 56.

28 ZIV, J.; LEMPEL, A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, v. 23, n. 3, p. 337–343, 1977. Citado nas páginas 55 e 56.

29 ZIV, J.; LEMPEL, A. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, v. 24, n. 5, p. 530–536, 1978. Citado nas páginas 55 e 56.

30 CLEARY, J.; WITTEN, I. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, v. 32, n. 4, p. 396–402, 1984. Citado na página 55.

31 NEVILL-MANNING, C.; WITTEN, I. Linear-time, incremental hierarchy inference for compression. In: *Proceedings DCC '97. Data Compression Conference*. [S.l.: s.n.], 1997. p. 3–11. Citado na página 55.

32 ARZ, J.; FISCHER, J. Lz-compressed string dictionaries. In: *2014 Data Compression Conference*. [S.l.: s.n.], 2014. p. 322–331. Citado na página 55.

33 CONRAD, K. J.; WILSON, P. R. Grammatical ziv-lempel compression: Achieving ppm-class text compression ratios with lz-class decompression speed. In: *2016 Data Compression Conference (DCC)*. [S.l.: s.n.], 2016. p. 586–586. Citado na página 55.

34 LIU, W. et al. Data compression device based on modified lz4 algorithm. *IEEE Transactions on Consumer Electronics*, v. 64, n. 1, p. 110–117, 2018. Citado na página 55.

35 TAJUL, T. K.; BHUIYAN, S. R.; HABIB, A. Enhancement of lzap (lempel ziv all prefixes) compression algorithm. In: *2018 4th International Conference on Electrical Engineering and Information Communication Technology (iCEEiCT)*. [S.l.: s.n.], 2018. p. 69–73. Citado na página 55.

36 WU, C. Y. Improved lz77 compression. In: *2021 Data Compression Conference (DCC)*. [S.l.: s.n.], 2021. p. 377–377. Citado na página 55.

37 OSWALD, C.; SIVASELVAN, B. An optimal text compression algorithm based on frequent pattern mining. *Journal of Ambient Intelligence and Humanized Computing*, v. 9, n. 3. Citado na página 55.

38 BREHM, M.; THOMAS, M. An efficient lossless compression algorithm for trajectories of atom positions and volumetric data. *Journal of Chemical Information and Modeling*, v. 58, n. 10, p. 2092–2107, 2018. Citado na página 55.

39 SHRIVIDHIYA, G. et al. Robust data compression algorithm utilizing lzw framework based on huffman technique. In: *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*. [S.l.: s.n.], 2021. p. 234–237. Citado na página 55.

40 REZNIK, Y. Coding of sets of words. In: *2011 Data Compression Conference*. [S.l.: s.n.], 2011. p. 43–52. Citado na página 56.

41 REZNIK, Y. Codes for unordered sets of words. In: *2011 IEEE International Symposium on Information Theory Proceedings*. [S.l.: s.n.], 2011. p. 1322–1326. Citado na página 56.

42 LEMPEL, A.; ZIV, J. On the complexity of finite sequences. *IEEE Transactions on Information Theory*, v. 22, n. 1, p. 75–81, 1976. Citado nas páginas 56 e 57.

43 COVER, T. M.; THOMAS, J. A. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006. ISBN 0471241954. Citado nas páginas 57, 65, 93, 95, 96, 97 e 100.

44 JALBERT, N.; SEN, K. A trace simplification technique for effective debugging of concurrent programs. In: *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2010. (FSE '10), p. 57–66. ISBN 9781605587912. Disponível em: <https://doi.org/10.1145/1882291.1882302>. Citado na página 71.

45 CORNELISSEN, B.; ZAIDMAN, A.; DEURSEN, A. van. A controlled experiment for program comprehension through trace visualization. *IEEE Transactions on Software Engineering*, v. 37, n. 3, p. 341–355, 2011. Citado na página 71.

46 SILVA, A. da; ASSIS, F. M. de. A new algorithm for compression of partially commutative alphabets. *Information Sciences*, v. 611, p. 107–125, 2022. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025522009409>. Citado na página 76.

47 SILVA, A. da; ASSIS, F. de. Novos algoritmos para a compressão de sequências parcialmente comutativas. In: *Anais do XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*. [S.l.: s.n.], 2022. Citado na página 76.

48 JUNGNICKEL, D. *Graphs, Networks and Algorithms*. 3rd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2007. ISBN 3540727795. Citado na página 83.

49 DIESTEL, R. *Graph Theory*. 5th. ed. [S.l.]: Springer Publishing Company, Incorporated, 2017. ISBN 3662536218. Citado na página 83.

50 STANLEY, R. P. *Enumerative Combinatorics: Volume 1*. 2nd. ed. USA: Cambridge University Press, 2011. ISBN 1107602629. Citado na página 85.

51 DILWORTH, R. P. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, Annals of Mathematics, v. 51, n. 1, p. 161–166, 1950. ISSN 0003486X. Disponível em: <http://www.jstor.org/stable/1969503>. Citado na página 86.

52 GALVIN, F. A proof of dilworth's chain decomposition theorem. *The American Mathematical Monthly*, Taylor & Francis, v. 101, n. 4, p. 352–353, 1994. Disponível em: <https://doi.org/10.1080/00029890.1994.11996954>. Citado na página 86.

53 MIRSKY, L. A dual of dilworth's decomposition theorem. *The American Mathematical Monthly*, Mathematical Association of America, v. 78, n. 8, p. 876–877, 1971. ISSN 00029890, 19300972. Disponível em: <http://www.jstor.org/stable/2316481>. Citado na página 87.

54 BRIGHTWELL, G.; WINKLER, P. Counting linear extensions is #p-complete. In: *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 1991. (STOC '91), p. 175–181. ISBN 0897913973. Disponível em: <https://doi.org/10.1145/103418.103441>. Citado na página 87.

55 MORALES, A. H.; PAK, I.; TASSY, M. Asymptotics for the number of standard tableaux of skew shape and for weighted lozenge tilings. *Combinatorics, Probability and Computing*, Cambridge University Press, p. 1–24, 2021. Citado na página 88.

56 WILF, H. S. *Algorithms and complexity (2. ed.)*. [S.l.]: A K Peters, 2002. ISBN 978-1-56881-178-9. Citado na página 98.

57 LI, M.; VITNYI, P. M. *An Introduction to Kolmogorov Complexity and Its Applications*. 3. ed. [S.l.]: Springer Publishing Company, Incorporated, 2008. ISBN 0387339981. Citado na página 100.

APÊNDICE A

Teoria dos Grafos

Os conceitos de grafos são fundamentais para o desenvolvimento dessa dissertação. Por isso, este capítulo é dedicado às definições básicas e às propriedades de grafos. Para mais definições, o leitor interessado pode consultar [48, 49].

Um *grafo simples não-orientado* G é um par $G = (V, E)$ que consiste em um conjunto V de vértices e um conjunto de arestas $E \subseteq V \times V$. Em G não são permitidas múltiplas arestas ligando dois vértices e nem arestas saindo e chegando no mesmo vértice. Dada uma aresta $e = (a, b)$ de E, é dito que a e b são adjacentes. Já pares de vértices não conectados por uma aresta são chamados de não-adjacentes. O complemento $\overline{G} = (V, \overline{E})$ de um grafo $G = (V, E)$ é obtido mantendo os mesmos vértices, mas com $\overline{E} = (V \times V) \setminus E$. O conjunto de vértices de um grafo G será referido como $V(G)$ e o conjunto de arestas como $E(G)$.

Um grafo em que todos os vértices são adjacentes par a par é chamado de *completo* e é denotado por K_n, em que n é o número vértices. Um grafo $G = (V, E)$ é chamado de r-partido se V admite uma partição em r subconjuntos disjuntos tais que cada vértice de uma partição é adjacente a um elemento de outro subconjunto, mas é não-adjacente aos vértices do subconjunto ao qual pertence. Quando cada elemento dos subconjuntos de um grafo r-partido é adjacente aos outros vértices, o grafo é chamado de r-partido completo e é denotado por $K_{n_1,n_2,\ldots,n_r}$, em que n_i corresponde ao número de elementos na i-ésima partição.

Na Fig. A.1 é apresentado um exemplo de representação de grafos. Cada vértice é representado por um quadrado com seu rótulo dentro. As arestas são representadas por uma linha ligando dois vértices. Neste exemplo, o conjunto de vértices e arestas são $V = \{a,b,c,d\}$ e $E = \{(a,b), (a,c), (a,d), (b,c), (b,d), (c,d)\}$, respectivamente, caracterizando um grafo K_4 completo.

Um *conjunto independente* ou conjunto estável de um grafo $G = (V, E)$ é um subconjunto de V em que nenhum dos vértices são adjacentes par a par. Um *clique* é um

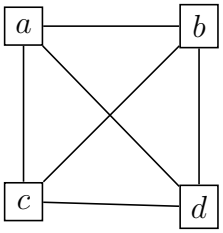


Figura A.1 – Grafo de K_4 completo.

subconjunto de V em que todos os vértices são adjacentes par a par. Os cliques e conjuntos independentes estão relacionados pelo fato de que um clique de G é um conjunto independente em $\overline{G}$. O *número do clique* de um grafo é a cardinalidade do maior clique desse grafo. Por exemplo, no grafo K_4 da Fig. A.1 o maior clique tem tamanho 4, sendo esse também o seu número do clique.

O problema de enumerar cliques (e conjuntos independentes) em grafos é NP-Completo (ver Apêndice D). Como não há uma solução eficiente (com tempo polinomial) para esses problemas, torna-se exponencialmente mais custoso temporalmente encontrar os cliques conforme os grafos crescem. Esse é um problema a ser considerado, pois a enumeração das classes de equivalência tratadas nesta dissertação depende da contagem do número de cliques. Por ser um problema difícil, um dos resultados dessa dissertação utiliza limitantes para o número máximo de cliques possíveis a fim de aproximar o número de classes de equivalência.

APÊNDICE B

Conjuntos Parcialmente Ordenados

Definição B.1 (POSET [50]). *Um conjunto parcialmente ordenado (POSET) P é formado por um conjunto não-vazio $\mathcal{P}$ aliado a uma relação binária $\leq_P$ que satisfaz as condições:*

- *Para todo $r \in P$, $r \leq_P r$ (Reflexividade);*
- *Para $r, s \in P$, $r \leq_P s$ e $s \leq_P r$ implica que $r = s$ (Anti-simetria); e*
- *Para $r, s \in P$, $r \leq_P s$ e $s \leq_P t$ implica $r \leq_P t$ (Transitividade).*

Satisfazendo essas condições, o POSET pode ser representado pela dupla $P = (\mathcal{P}, \leq_P)$.

Será utilizado $\leq$ ao invés de $\leq_P$ quando for conveniente. Dois elementos $r, s \in P$ são comparáveis se $r \leq s$ ou $r \leq s$ e são incomparáveis, caso contrário. Um subconjunto é totalmente ordenado se todos os pares de elementos desse subconjunto são comparáveis. Se $r, s \in P$ e $r \leq s$, diz-se que s *cobre* r. Um POSET pode ser representado por um grafo chamado de diagrama de Hasse.

Definição B.2 (Diagrama de Hasse). *Um diagrama de Hasse que representa um POSET $P = (\mathcal{P}, \leq)$ é um grafo direcionado formado pelos elementos de $\mathcal{P}$. As arestas são definidas pelas relações de cobertura de P, de tal forma que se $r, s \in P$ e $r \leq s$, então s aparece acima de r no grafo.*

Considere, como exemplo, o POSET formado pelo subconjunto $\mathcal{P} = \{1,3,5,6,15,12,35,80\}$ e a relação é a divisibilidade ($|$). Em outras palavras, se $a|b$ (lê-se se a divide b), então $a \leq b$. O diagrama de Hasse obtido por meio desse POSET é apresentado na Fig. B.1.

Cabe notar que, apesar de $3|12$, não há uma aresta ligando 3 e 12. Isso se repete para relações redundantes, pois não há necessidade de representar a aresta. Como o elemento

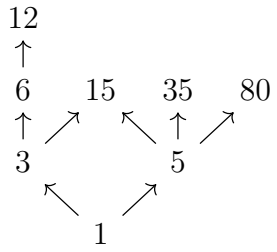


Figura B.1 – Exemplo de diagrama de Hasse do POSET da relação de divisibilidade.

1 está abaixo de todos, ele é chamado de *elemento mínimo* de P. Como não há nenhum elemento acima ao lado de 12, ele é o *elemento máximo* de P. Outros conceitos necessários são apresentados a seguir.

Definição B.3 (Cadeia e Anti-Cadeia)**.** *Seja P um POSET. Uma cadeia é um subconjunto totalmente ordenado de $\mathcal{P}$,i.e., em que todos os pares são comparáveis. Uma anti-cadeia é um subconjunto de $\mathcal{P}$ em que qualquer par de nós é incomparável.*

Definição B.4 (Cadeia e Anti-Cadeia Máxima)**.** *Uma cadeia é chamada de máxima se não estiver contida em maiores cadeias. Uma anti-cadeia é chamada de máxima se não estiver contida em maiores anti-cadeias.*

Definição B.5 (Altura e largura de um POSET)**.** *A altura de um POSET é a cardinalidade da maior cadeia e a largura é a cardinalidade da maior anti-cadeia.*

Considerando o POSET da Fig. B.1, exemplos de cadeias de P são $\{1,3,6,12\}, \{5, 15\}, \{1\}$, etc. Exemplos de anti-cadeias são $\{3, 5\}, \{6, 15, 35, 80\}, \{1\}$, etc. Nota-se que cadeias e anti-cadeias podem ser formadas por um único elemento. A cadeia máxima de P é $\{1,3,6,12\}$ sendo sua altura igual a 4. Uma anti-cadeia máxima de P é $\{6, 15, 35, 80\}$ (outra anti-cadeia máxima é $\{12, 15, 35, 80\}$), então a largura de P também é 4.

Definição B.6 (Nível de um elemento do POSET)**.** *Seja $P = (\mathcal{P}, \leq)$ um POSET e $\rho(r)$ chamada de função de nível que retorna o nível do elemento $r \in P$. Para dois elementos $r, s, \in P$, se $r \leq s$, então $\rho(s) = \rho(r) + 1$. Além disso $\rho(x) = 0$, sendo x um elemento mínimo de P.*

A definição de nível de um elemento está relacionado à altura em que ele se encontra no diagrama de Hasse. Desta forma, para o POSET da Fig. B.1, tem-se que $\rho(1) = 0$, $\rho(3) = \rho(5) = 1$, $\rho(6) = \rho(15) = \rho(35) = \rho(80) = 2$ e $\rho(12) = 3$.

Teorema B.1 (Teorema de Dilworth [51, 52])**.** *Qualquer POSET P finito de largura w pode ser particionado em w cadeias disjuntas, ou ainda, $\mathcal{P} = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_w$.*

Teorema B.2 (Teorema dual de Dilworth [53]). *Qualquer POSET finito de altura h pode ser particionado em h anti-cadeias disjuntas, ou ainda,* $\mathcal{P} = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_h$.

Ainda considerando o POSET da Fig. B.1, sabe-se que sua largura é igual a $w = 4$ e sua altura é igual a $h = 4$. Desta forma, P pode ser particionado em 4 cadeias disjuntas (*e.g.*, $\mathcal{P} = \{1, 3, 6, 12\} \cup \{5, 15\} \cup \{35\} \cup \{80\}$, $\mathcal{P} = \{1, 5, 80\} \cup \{3, 6, 12\} \cup \{15\} \cup \{35\}$) e em 4 anti-cadeias disjuntas (*e.g.*, $\mathcal{P} = \{12, 15, 35, 80\} \cup \{6, 5\} \cup \{3\} \cup \{1\}$, $\mathcal{P} = \{6, 15, 35, 80\} \cup \{3, 5\} \cup \{12\} \cup \{1\}$). Pode-se notar que a partição não é necessariamente única.

Definição B.7 (Extensão Linear). *Seja um POSET* $P = (\mathcal{P}, \leq_P)$. *Uma extensão linear de P é uma permutação* $p_{\pi(1)}p_{\pi(2)} \cdots p_{\pi(n)}$ *dos elementos* $p_1p_2 \cdots p_n$ *de* $\mathcal{P}$ *tal que* $p_{\pi(i)} \leq_P p_{\pi(j)}$ *implica em* $i < j$, *sabendo que* $\pi(i)$ *é o mapeamento do índice i em outro.*

Uma definição equivalente para extensão linear de um POSET P é encontrar uma bijeção λ de P em $\{1, \ldots, n\}, n = |\mathcal{P}|$, tal que $\lambda(r) < \lambda(s)$ sempre que $r \leq_P s$, para todos os $r, s \in P$ [54].

Considere o POSET P da Fig. B.2. As extensões lineares devem ser tais que as relações $1 \leq_P 3$ e $2 \leq_P 4$ devem ser mantidas. Desta forma, o índice do elemento 3 na permutação deve ser sempre maior do que o índice do elemento 1. Analogamente, o índice do elemento 3 deve ser sempre maior do que o do elemento 2 na permutação. Um exemplo de extensão linear é $p_{\pi(1)}p_{\pi(2)}p_{\pi(3)}p_{\pi(4)} = 1324$, pois $1 \leq_P 3 \implies 1 < 2$ e $2 \leq_P 4 \implies 3 < 4$. Outro exemplo de extensão linear é $p_{\pi(1)}p_{\pi(2)}p_{\pi(3)}p_{\pi(4)} = 1243$, pois $1 \leq_P 3 \implies 1 < 4$ e $2 \leq_P 4 \implies 2 < 3$. Seguindo o mesmo raciocínio, obtém-se as extensões lineares possíveis de P que são $1234, 1243, 1324, 2134, 2143, 2413$.

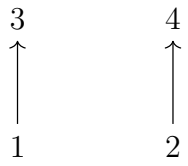


Figura B.2 – Diagrama de Hasse para exemplificar extensão linear.

Tratando do problema de contar o número de extensões lineares, Brightwell e Winkler [54] demonstraram o Teorema B.3 acerca da complexidade computacional de determinar o número de extensões lineares de um POSET.

Teorema B.3 ([54]). *Contar o número de extensões lineares* $e(P)$ *de um POSET é um problema #P-completo.*

Um problema #P-completo (lido como P-número completo) é tão difícil quanto um problema NP-Completo, de forma que um algoritmo com tempo polinomial que resolva

um problema #P-completo também responderia um dos problemas do milênio das classes P e NP. Entretanto, nenhum algoritmo com complexidade polinomial que resolva um problema #P-completo é conhecido. Apesar de ser um problema difícil, ainda é possível obter limitantes para o número de extensões lineares de um POSET [55].

Teorema B.4 ([17])**.** *Considere a decomposição de um POSET $P = (\mathcal{P}, \leq)$ em k cadeias $C_1, \ldots, C_k$ e l anti-cadeias $A_1, \ldots, A_l$, tais que $c_i = |C_i|$, $n = c_1 + \cdots + c_k$, $a_i = |A_i|$, $n = a_1 + \cdots + a_l$ e $n = |P|$, então o número $e(P)$ de extensões lineares obedece às desigualdades*

$$\prod a_i! \leq e(P) \leq \frac{n!}{\prod c_i!}. \tag{B.1}$$

Os valores de c_i e a_i do Teorema B.4 podem ser obtidos por meio do algoritmo da resolução do problema de fluxo máximo em grafos orientados [16, 15]. Para isso, é utilizado o Algoritmo de Ford-Fulkerson.

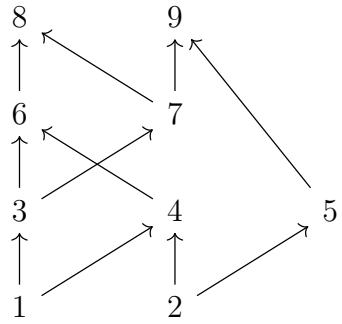
B.1 Descrição do Algoritmo de Ford-Fulkerson

Os valores de c_i e a_i podem ser obtidos por meio do algoritmo da resolução do problema de fluxo máximo em grafos orientados [16, 15].

Seja $P = (\mathcal{P}, \leq_P)$ um POSET e M uma matriz com dimensões $n \times n$, em que $n = |\mathcal{P}|$. Seja também $i = 1, \ldots, n$ os índices das linhas e $j = 1, \ldots, n$ os índices das colunas da matriz M, de forma que cada linha e cada coluna está associada a um elemento de $\mathcal{P}$. Se o elemento associado à linha i é comparável, de acordo com $\leq_P$, ao elemento associado à coluna j, então a entrada (i, j) da matriz é chamada de *célula admissível*, sendo representada graficamente como uma célula vazia. A célula admissível pode receber o número 1, significando que o elemento da linha é conectado ao elemento da coluna por uma aresta. Caso o elemento associado à linha i não seja comparável de acordo com $\leq_P$ ao elemento associado à coluna j, então a entrada (i, j) da matriz é chamada de célula *não-admissível*, que é representada por uma célula contendo $\circ$ e não pode receber valores. Nessas condições, a matriz M é chamada de *matriz admissível*.

Na Fig. B.3 são apresentados, como exemplo, o diagrama de Hasse de um POSET (Fig.B.3a) e sua matriz admissível (B.3b). Esse exemplo será utilizado para explicar o algoritmo proposto por Ford e Fulkerson [15] que será doravante chamado de *Algoritmo FF*.

Seguindo a definição da matriz admissível, espera-se que a diagonal seja formada somente por células não-admissíveis, pois um elemento não é comparável a si mesmo. Como o elemento 1 não é comparável a 2 e 5, então para a linha 1, nas colunas 2 e 5



(a) Diagrama de Hasse de P.

	1	2	3	4	5	6	7	8	9
1	○	○			○				
2	○	○	○				○		
3	○	○	○	○	○				
4	○	○	○	○	○		○		○
5	○	○	○	○	○	○	○	○	
6	○	○	○	○	○	○	○		○
7	○	○	○	○	○	○	○		
8	○	○	○	○	○	○	○	○	○
9	○	○	○	○	○	○	○	○	○

(b) Matriz admissível.

Figura B.3 – Diagrama de Hasse e matriz admissível associada.

estão presentes células não admissíveis (○). O mesmo raciocínio é utilizado para construir o restante da matriz admissível.

A solução para o problema de fluxo máximo no POSET é obtida colocando o maior número de 1 possíveis nas células admissíveis seguindo a restrição de que no máximo um número 1 pode ser colocado em uma linha e em uma coluna. Desta forma, o passo inicial do algoritmo é percorrer as linhas em ordem crescente e colocar 1 na primeira célula admissível da linha atual que está em uma coluna ainda sem 1. Seguindo esse processo, obtém-se a disposição de 1 representada na Fig. B.4a. A segunda etapa é rotular as linhas que não possuem 1 usando ◁, como representado na Fig. B.4b.

	1	2	3	4	5	6	7	8	9
1	○	○	1		○				
2	○	○	○	1			○		
3	○	○	○	○	○	1			
4	○	○	○	○	○		○	1	○
5	○	○	○	○	○	○	○	○	1
6	○	○	○	○	○	○	○		○
7	○	○	○	○	○	○	○		
8	○	○	○	○	○	○	○	○	○
9	○	○	○	○	○	○	○	○	○

(a) Matriz admissível com a disposição inicial de 1.

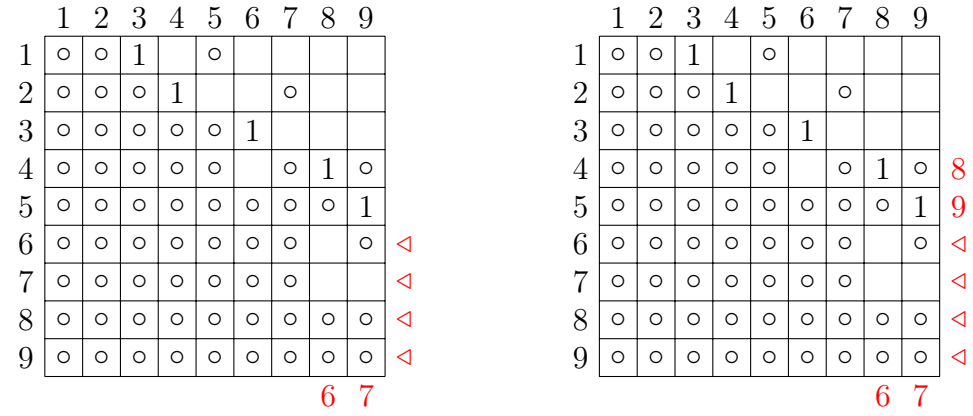
	1	2	3	4	5	6	7	8	9	
1	○	○	1		○					
2	○	○	○	1			○			
3	○	○	○	○	○	1				
4	○	○	○	○	○		○	1	○	
5	○	○	○	○	○	○	○	○	1	
6	○	○	○	○	○	○	○		○	◁
7	○	○	○	○	○	○	○			◁
8	○	○	○	○	○	○	○	○	○	◁
9	○	○	○	○	○	○	○	○	○	◁

(b) Rotulação inicial das linhas com sem 1 usando ◁.

Figura B.4 – Marcação inicial da matriz admissível associada.

Para cada uma das linhas rotuladas, se a linha i (rotulada) possui células admissíveis nas colunas $j_1, j_2, \ldots$, então essas colunas recebem o rótulo i. Esse processo será referido como *rotulação das colunas*. Além disso, se uma coluna já foi rotulada, não se deve adicionar um novo rótulo. Lembrando que no exemplo as linhas rotuladas com ◁ são as 6, 7, 8 e 9. Começando pela linha $i = 6$, então a coluna 8 é rotulada com 6. Como não

há outras células admissíveis na linha 6, então verifica-se as células admissíveis da linha 7, que estão nas colunas 8 e 9. Como a coluna já foi rotulada, então somente a coluna 9 recebe o rótulo 7. Como as linhas 8 e 9 não possuem células admissíveis, então essa etapa termina. O resultado é apresentado na Fig. B.5a. Para cada uma das colunas rotuladas, verifica-se se há 1, de forma que se a coluna j possuir um 1 na linha i, então esta linha recebe j como seu rótulo. Esse processo será referido como *rotulação das linhas*. Desta forma, as colunas rotuladas são 8 e 9, então verifica se na coluna 8 há 1. O 1 da coluna 8 está na linha 4 que é rotulada com 8. Repetindo o processo para a coluna 9, a linha 5 é rotulada com 9. Desta forma, as novas linhas rotuladas são as linhas 4 e 5 e a matriz resultante é apresentada na Fig. B.5b.



(a) Rotulação das colunas 8 e 9. (b) Rotulação das linhas 4 e 5.

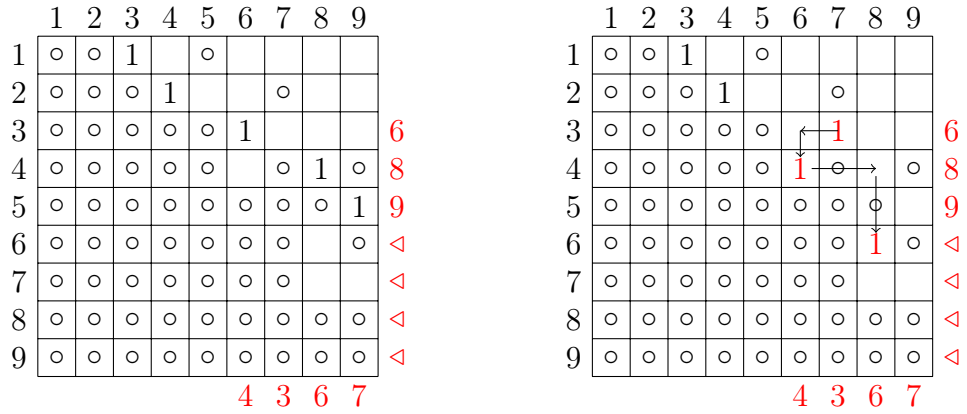
Figura B.5 – Rotulação das colunas e linhas da matriz admissível.

O processo de rotulação de linhas e colunas continua, alternadamente, até não ser possível rotular mais nenhuma linha, sendo duas as condições de parada:

1. Se uma coluna que não possui 1 é rotulada (inovação); ou
2. Se os rótulos gerados são referentes às linhas já rotuladas (fim).

Quando (2) ocorre o algoritmo para e se obtém a matriz admissível com o número máximo de 1, representando o fluxo máximo no grafo. Quando (1) ocorre, é possível aumentar o número de 1 da matriz em uma unidade. Quando a coluna j sem 1 for rotulada com i, então na entrada (i, j) adiciona-se um novo 1. Como na linha i é permitido somente um 1, então é necessário retirar o 1 anterior. O 1 anterior da linha i está entrada $(i, l_r(i))$, sendo $l_r(i)$ o rótulo da linha i. Como o 1 da coluna $l_r(i)$ foi retirado, então adiciona um 1 na entrada $(l_c(l_r(i)), l_r(i))$, sendo $l_c()$ o rótulo da coluna. Mais uma vez, o antigo 1 da linha $l_c(l_r(i))$ é removido, estando na entrada $(l_c(l_r(i)), l_r(l_c(l_r(i))))$. Esse processo continua até o 1 ser adicionado a uma linha rotulada com $\triangleleft$. Após o processo da situação (1), todo processo de rotulação é repetido e a situação (2) ocorre.

Dando continuidade ao exemplo, é realizada novamente a rotulação das colunas considerando agora as linhas 4 e 5, obtém-se que a coluna 6 é rotulada com 4. Como a linha 5 não rotula nenhuma coluna, então o processo continua para a rotulação das linhas. Para a coluna 6, o 1 está presente na linha 3 que é rotulada com 6. A partir da linha 3, a coluna 7 é rotulada. Como não existe 1 na coluna 7, então não é possível rotular mais nenhuma linha e a matriz resultante é apresentada na Fig. B.6a. Desta forma, a situação (2) ocorre. Como a coluna rotulada é a 7 e o rótulo dessa coluna é 3, então na entrada $(3, 7)$ é adicionado 1 e em $(3, l_r(3)) = (3, 6)$ é retirado o 1 anterior da linha e assim por diante. O processo de atualização da disposição de 1 na matriz é representado na Fig. B.6b, em que os novos 1 estão destacados e as setas indicam o sentido das substituições.



(a) Rotulação da coluna 7 que não possui 1.

(b) Processo de atualização das células com número 1.

Figura B.6 – Rotulação da matriz admissível nos estágios finais.

Um novo processo de rotulação é realizado de forma que o resultados é uma matriz admissível com um 1 a mais como apresentado na Fig. B.7a. Após o processo de inovação, a situação (1) ocorre e na Fig B.7b é representado o diagrama de Hasse contendo somente um conjunto de cadeias máximas do POSET inicial.

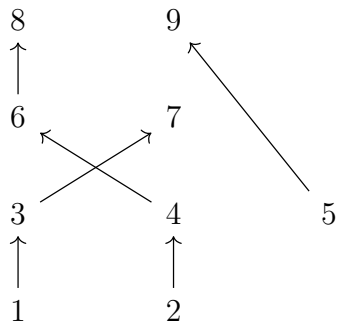
Usando a Fig B.7b, é possível concluir que {2,4,6,8}, $\{1, 3, 7\}$ e $\{5, 9\}$ é uma decomposição do POSET inicial em cadeias máximas, sendo possível definir os valores dos parâmetros do limite superior do Teorema B.4, sendo elas $c_1 = |\{2,4,6,8\}| = 4$, $c_2 = |\{1, 3, 7\}| = 3$ e $c_3 = |\{5, 9\}| = 2$.

Os parâmetros a_i, por sua vez, podem ser obtidas por meio do algoritmo que faz a identificação e retirada sucessiva dos elementos mínimos de um POSET. Esse procedimento corresponde ao Algoritmo 4.

Para exemplificar o Algoritmo 4, será utilizado o POSET da Fig. B.8. Os elementos mínimos inicialmente são $A_1 = \{1, 2\}$. Ao desconsiderar os elementos de A_1, os novos elementos mínimos são $A_2 = \{3, 4, 5\}$. Desconsiderando os elementos de A_1 e A_2, os elementos mínimos são $A_3 = \{6, 7\}$ e, por fim, $A_4 = \{8, 9\}$.

	1	2	3	4	5	6	7	8	9	
1	○	○	1		○					
2	○	○	○	1			○			
3	○	○	○	○	○		1			
4	○	○	○	○	○	1	○		○	
5	○	○	○	○	○	○	○	○	1	9
6	○	○	○	○	○	○	○	1	○	8
7	○	○	○	○	○	○	○			◁
8	○	○	○	○	○	○	○	○	○	◁
9	○	○	○	○	○	○	○	○	○	◁
								7	7	

(a) Matriz admissível após a conclusão do Algoritmo FF.



(b) Diagrama de Hasse contendo as cadeias máximas.

Figura B.7 – Estado final da matriz admissível e cadeias máximas.

Algoritmo 4: Decomposição em Anti-cadeias

Entrada: POSET $P = (\mathcal{P}, \leq)$
Saída: $A = (A_1, \ldots, A_l)$

$\mathcal{Q} \leftarrow \mathcal{P}$
$A \leftarrow \emptyset$ ▷ *Conjunto das anti-cadeias*
enquanto $|Q| \neq 0$ **faça**
  $A_i \leftarrow$ elementos mínimos de $(\mathcal{Q}, \leq)$
  $A \leftarrow A \cup \{A_i\}$
  $\mathcal{Q} \leftarrow \mathcal{Q} - A_i$

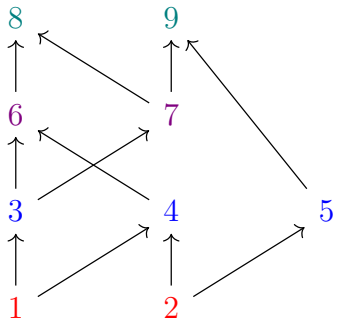


Figura B.8 – Diagrama de Hasse do POSET P.

Os elementos de A_i são incomparáveis e, portanto, que A_i corresponde a uma anti-cadeia. Considerando o exemplo, os parâmetros do limite inferior do Teorema B.4 serão $a_1 = |A_1| = 2$, $a_2 = |A_2| = 3$, $a_3 = |A_3| = 2$, $a_4 = |A_4| = 2$. Desta forma, é possível utilizar o Teorema B.4 para definir os limitantes para o número $e(P)$ de extensões lineares do POSET P como

$$2!3!2!2! \leq e(P) \leq \frac{9!}{4!3!2!} \implies 48 \leq e(P) \leq 1260. \tag{B.2}$$

APÊNDICE C

Teoria da Informação

C.1 Entropia de Shannon

Como mencionado, o conceito de entropia foi introduzido por Shannon [1] e refere-se à medida de incerteza de uma variável aleatória.

Definição C.1 (Entropia)**.** *Seja X uma variável aleatória discreta com alfabeto $\mathcal{X}$ e densidade de probabilidade $p(x) = \Pr[X = x]$, $x \in \mathcal{X}$. A* entropia $H(X)$ *da variável discreta é definida por*

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{C.1}$$

Ao longo desta dissertação, se utilizará a convenção de que $0 \log 0 = 0$, justificado por $\lim_{x \to 0} x \log x = 0$. Se a base do logaritmo é b, denotaremos a entropia como $H_b(X)$. Se a base do logaritmo é e, a entropia é medida em *nats*.

C.2 Propriedade de Equipartição Assintótica

A propriedade de equipartição assintótica (AEP) possibilita a classificação das sequências de variáveis aleatórias (va) em dois conjuntos, as *sequências típicas* e as *sequências não-típicas*. As sequências típicas são aquelas que possuem as entropias empíricas próximas à entropia da fonte e as sequências não-típicas são aquelas que não possuem essa propriedade. A propriedade da equipartição assintótica (AEP) é apresentada no Teorema C.1.

Teorema C.1. *(AEP [43, p.58])*

Sejam $X_1, X_2, \ldots, X_n$ v.a. independentes e identicamente distribuídas (iid) com distribuição p, então

$$-\frac{1}{n} \log_2 p\left(X_1, X_2, \ldots, X_n\right) \to H(X) \tag{C.2}$$

em probabilidade. Ou seja, para todo $\epsilon > 0$ *vale*

$$\Pr\left\{\left|-\frac{1}{n}\log_2 p\left(X_1, X_2, \ldots, X_n\right) - H(X)\right| > \epsilon\right\} \to 0.$$

Uma das interpretações para as sequências típicas é que elas são quase igualmente surpreendentes, no sentido de apresentarem uma incerteza próxima à da entropia da fonte. A partir da AEP, é possível definir o chamado de conjunto típico que é formado por todas as sequências de comprimento n que apresentam uma probabilidade compreendida em um intervalo definido.

Definição C.2. *(Conjunto Típico) O conjunto típico* $A_\epsilon^{(n)}$, $\epsilon > 0$ *com respeito à* $p(x)$ *é o conjunto das sequências* $(x_1, x_2, \ldots, x_n) \in \Sigma^n$ *com a propriedade*

$$2^{-n(H(X)+\epsilon)} \leq p(x_1, x_2, \ldots, x_n) \leq 2^{-n(H(X)-\epsilon)} \tag{C.3}$$

em que Σ *é o alfabeto de símbolos.*

Algumas das propriedades do conjunto típico são apresentadas no Teorema C.2.

Teorema C.2. *(Propriedades do Conjunto Típico)*

1. *Se* $(x_1, x_2, \ldots, x_n) \in A_\epsilon^{(n)}$, *então* $H(X) - \epsilon \leq -\frac{1}{n}\log_2 p(x_1, x_2, \ldots, x_n) \leq H(X) + \epsilon$.
2. $\Pr\{A_\epsilon^{(n)}\} > 1 - \epsilon$;
3. *A cardinalidade do conjunto típico* $A_\epsilon^{(n)}$ *tem como limite superior* $\left|A_\epsilon^{(n)}\right| \leq 2^{n(H(X)+\epsilon)}$;
4. *A cardinalidade do conjunto típico* $A_\epsilon^{(n)}$ *tem como limite inferior* $\left|A_\epsilon^{(n)}\right| \geq (1-\epsilon)\, 2^{n(H(X)-\epsilon)}$

para n grande o suficiente.

A primeira propriedade segue diretamente da definição de conjunto típico. A segunda propriedade indica que pode-se definir o conjunto típico de forma que as sequências presentes nesse conjunto apareçam com uma probabilidade arbitrariamente próxima a 1. A terceira propriedade indica que a cardinalidade do conjunto típico é menor ou igual à cardinalidade do conjunto total de sequências de comprimento definidas sobre um alfabeto finito Σ que é igual a $|\Sigma|^n$.

Em outras palavras, existe um conjunto de sequências que ocorrem com alta probabilidade. Além disso, a cardinalidade desse conjunto é menor do que a cardinalidade do

conjunto de todas as sequências, fazendo com que sejam necessários $2^{n(H+\epsilon)}$ *bits* em média para representá-lo, como exemplificado na Fig. C.1. Essas propriedades do conjunto típico podem ser utilizadas para a compressão com perdas de sequências.

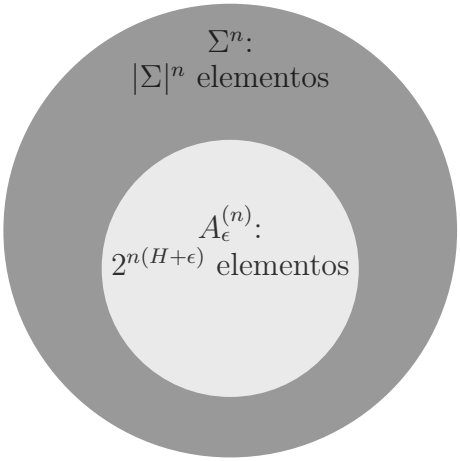


Figura C.1 – Conjunto das sequências e conjunto típico.

O Teorema C.3 apresenta que a entropia $H(X)$ é o limite inferior do número de *bits* necessários para representar símbolos de uma sequência com certa distribuição de probabilidade.

Teorema C.3. *([43, p.62]) Seja $X^n = X_1, X_2, \ldots, X_n$ uma sequência iid com distribuição $p(x)$ e seja $\epsilon > 0$, então existe um código que mapeia a sequência $x^n = x_1, x_2, \ldots, x_n$ de comprimento n em palavras binárias tais que o mapeamento é inversível e*

$$\mathbb{E}\left[\frac{1}{n} l(X^n)\right] \leq H(X) + \epsilon \tag{C.4}$$

em que $l(X^n)$ denota o comprimento da palavra-código correspondente a X^n. Em outras palavras, é possível representar as sequências X^n usando $nH(X)$ bits em média.

C.3 Teoria dos Tipos

Enquanto a AEP possibilita a compressão por demonstrar que existe um subconjunto das sequências possíveis que apresenta alta probabilidade, a Teoria dos Tipos utiliza semelhanças entre as sequências para agrupar nos chamados de tipos. Esses tipos estão relacionados à frequência relativa de ocorrência de símbolos de um alfabeto nessas sequências.

Definição C.3. *(Tipos) O tipo $P_{\mathbf{x}}$ (ou distribuição empírica de probabilidade) de uma sequência $x_1, x_2, \ldots, x_n$ é a frequência de ocorrência relativa de cada símbolo do alfabeto Σ. Desta forma, para todo $a \in \Sigma$, tem-se $P_{\mathbf{x}}(a) = N(a|\mathbf{x})/n$, em que $N(a|\mathbf{x})$ denota o número de ocorrências do símbolo a na sequência $\mathbf{x} \in \Sigma^n$.*

Definição C.4. *(Conjunto dos Tipos com Denominador n) O conjunto dos tipos com denominador n é formado pelas $|\Sigma|$-tuplas de frequências relativas possíveis considerando uma sequência $x_1, x_2, \ldots, x_n$ e é denotado por $\mathcal{P}_n$.*

Exemplo C.1. *(Exemplo de Conjunto dos Tipos com Denominador n) Seja $\Sigma = \{0, 1\}$ o conjunto dos tipos com denominador n é*

$$\mathcal{P}_n = \left\{ (P(0), P(1)) : \left(\frac{0}{n}, \frac{n}{n}\right), \left(\frac{1}{n}, \frac{n-1}{n}\right), \ldots, \left(\frac{n}{n}, \frac{0}{n}\right) \right\}$$

Definição C.5. *(Classe de Tipo) Se $P \in \mathcal{P}_n$, então o conjunto de sequências de comprimento n e classe P é chamado de classe de tipo P, denotada por $T(P)$ e*

$$T(P) = \{\mathbf{x} \in \Sigma^n : P_{\mathbf{x}} = P\} \tag{C.5}$$

Em outras palavras, a classe de tipo associada a um tipo $P_{\mathbf{x}}$ é formada por todas as sequências que apresentam a mesma distribuição empírica de probabilidade, i.e., o mesmo tipo de $P_{\mathbf{x}}$.

Exemplo C.2. *Seja $\Sigma = \{1, 2, 3\}$ e $\mathbf{x} = 123322$. Então, por definição, o tipo $P_{\mathbf{x}}$ é*

$$P_{\mathbf{x}}(1) = \frac{1}{6} \qquad P_{\mathbf{x}}(2) = \frac{3}{6} \qquad P_{\mathbf{x}}(3) = \frac{2}{6}$$

A classe de tipo $P_{\mathbf{x}}$ é formado pelas sequências de comprimento $n = 6$ com uma ocorrência de 1, três ocorrências de 2 e duas ocorrências de 3, ou seja,

$$T(P_{\mathbf{x}}) = \{123322, 123232, \ldots, 332221\}$$

em que o número de elementos de $T(P_{\mathbf{x}})$ é dado por

$$|T(P_{\mathbf{x}})| = \binom{6}{1, 2, 3} = 60$$

O número de elementos da classe de tipo pode ser calculado de forma geral por coeficientes multinomiais. O próximo teorema fornece um limite superior polinomial para o número de elementos do conjunto de tipos com denominador n.

Teorema C.4. *([43, p.349])*

$$|\mathcal{P}_n| \leq (n+1)^{|\Sigma|} \tag{C.6}$$

Enquanto o número de sequências cresce exponencialmente com n, o número de tipos distintos com denominador n é majorado por uma função polinomial de n. Desta forma, se referindo aos tipos ao invés das sequências individuais são necessários menos *bits* de informação para representar e construir sequências equivalentes.

Outra propriedade importante relacionada aos tipos é que a probabilidade de ocorrência de uma sequência de certo tipo só depende do seu tipo, como é apresentado no Teorema C.5 e no corolário C.1.

Teorema C.5. *([43, p.349]) Se $X_1, X_2, \ldots, X_n$ são iid e possuem distribuição $Q(x)$, a probabilidade da sequência depende somente do seu tipo e é dada por*

$$Q^n(\mathbf{x}) = \prod_{i=1}^{n} Q\left(x_i\right) = 2^{-n(H(P_{\mathbf{x}})+D(P_{\mathbf{x}}||Q))}, \tag{C.7}$$

em que $D\left(P_{\mathbf{x}}||Q\right)$ é a distância de Kullback-Leibler.

A partir do Teorema C.6 é possível associar o número de sequências com certo tipo à entropia.

Corolário C.1. *Se* $\mathbf{x}$ *está na classe de tipo Q, então*

$$Q^n(\mathbf{x}) = 2^{-nH(Q_{\mathbf{x}})}. \tag{C.8}$$

Teorema C.6. *(Tamanho de classe de tipo $T(P)$) Para qualquer tipo $P \in \mathcal{P}_n$, tem-se*

$$\frac{1}{(n+1)^{|\Sigma|}} 2^{nH(P)} \leq |T(P)| \leq 2^{nH(P)} \tag{C.9}$$

em que $H(P)$ se refere à entropia considerando a distribuição de probabilidade do tipo P.

Por consequência, o número de sequências de comprimento n que tem um tipo P vai sempre obedecer à desigualdade do Teorema C.6.

APÊNDICE D

Complexidade de Algoritmos e de Kolmogorov

D.1 Conceitos de Complexidade de Algoritmos

Nesta seção são abordados conceitos fundamentais de complexidade de algoritmos. Para mais detalhes, consultar [56]. A primeira definição acerca de complexidade necessária nessa proposta é a noção de que uma função não cresce mais rapidamente que outra. Essa noção é formalizada pela definição de Grande-O.

Definição D.1 (Notação Grande-O)**.** *Denota-se* $f(x) = O(g(x))$ *se existem constantes e e* x_0 *tais que* $|f(x)| < cg(x)$ *para todo* $x > x_0$*.*

Como exemplo do grande-O, sejam as funções $f(x) = x^4 + x^2\cos(x) + 2$ e $g(x) = x^5$. Considerando uma constante $c = 1$, tem-se $|f(x)| < g(x)$ para $x > x_0 = 2$. Consequentemente, pode-se escrever $x^4 + x^2\cos(x) + 2 = O(x^5)$. Em outras palavras, x^5 cresce mais rapidamente que $x^4 + x^2\cos(x) + 2$, para $c = 1$ e $x_0 = 2$, servindo como um limite superior.

Para definir classes de algoritmos, é necessário primeiro definir o que é um problema de decisão que pode receber um número infinito de entradas. Isso se deve porque a maioria dos problemas pode ser traduzida em termos de problemas de decisão. Cada entrada é chamada de *instância* do problema. Problemas de decisão são problemas que podem ser escritos como perguntas que aceitam *sim ou não* como resposta. Alguns exemplos de problemas de decisão são: x é um número primo? x divide y sem deixar resto?

Todo problema de decisão pode ser pensado como uma pergunta, se determinada sequência de entrada pertence a uma linguagem, sendo essa linguagem composta por todas as sequências de entrada que produzem "sim" como resposta.

Um problema de decisão pertence à *classe P* se existe um algoritmo $\mathcal{A}$ e uma constante c tal que para toda instância I do problema, $\mathcal{A}$ produzirá uma resposta em um tempo

$O(B^c)$, onde B é o número de bits da entrada que representa I. Em outras palavras, os problemas da classe P requerem algoritmos que produzem resultados em tempo polinomial (daí o "P") e são do tipo de problemas mais simples e é dito que possuem uma solução rápida.

A definição da classe NP não é tão simples quanto a da classe P. O termo NP está relacionado ao fato de que os algoritmos são de tempo polinomial não-determinístico. Um problema de decisão Q pertence à classe NP se existe um algoritmo $\mathcal{A}$ e

1. Associado a cada sequência da linguagem Q (que gera sim como resposta), existe um certificado $C(I)$ (uma justificativa para resposta "sim") que, quando $(I, C(I))$ é a entrada, então $\mathcal{A}$ retorna "sim";

2. Se I não pertence a Q, então não existe nenhum $C(I)$ tal que a entrada $(I, C(I))$ faça $\mathcal{A}$ retornar "sim";

3. O algoritmo $\mathcal{A}$ opera em um tempo polinomial.

Enquanto os problemas da classe P são fáceis de resolver, os da classe NP são *fáceis de verificar* se determinada instância pertence à linguagem com a ajuda de informação extra (nesse caso, o certificado). Em muitos casos, achar o certificado também é difícil.

Um problema Q é dito NP-completo se pertence à classe NP e todo problema NP pode ser expresso em termos de Q a partir de um procedimento que tem tempo polinomial. Essa expressão de um problema em termos de outro é chamado de redutibilidade. Uma consequência disso é que, se um problema NP-completo for solucionado, todos os outros da classe NP também serão. Alguns exemplos de problemas NP-completos são o de satisfatibilidade booleana (SAT), caxeiro viajante, coloração de grafos, problema do clique.

D.2 Complexidade de Kolmogorov

Nesta seção serão apresentadas as principais propriedades da complexidade de Kolmogorov. Serão considerados um alfabeto finito Σ e o conjunto de todas as palavras definidas sobre Σ denotado por Σ^*, incluindo o símbolo vazio ϵ.

Definição D.2. *(Complexidade de Kolmogorov) A complexidade de Kolmogorov $K_{\mathcal{U}}(\mathbf{u})$ da palavra $\mathbf{u} \in \Sigma^*$ com respeito a um computador universal $\mathcal{U}$ é definida como o menor comprimento $l(p)$ do programa p entre todos os programas que exibem u e param, i.e.,*

$$K_{\mathcal{U}}(\mathbf{u}) = \min_{p:\mathcal{U}(p)=\mathbf{u}} l(p). \tag{D.1}$$

Em outras palavras, a complexidade $K_{\mathcal{U}}(\mathbf{u})$ é a menor descrição de u entre as descrições possíveis de $\mathbf{u}$ *realizadas por um computador* $\mathcal{U}$*.*

A partir desse ponto as complexidades serão consideradas em relação a um mesmo computador universal $\mathcal{U}$, então será utilizando somente $K(\mathbf{u})$ para denotar a complexidade de Kolmogorov da palavra $\mathbf{u} \in \Sigma^*$.

A complexidade de Kolmogorov é subaditiva [57], i.e., a complexidade de duas palavras $\mathbf{u}$ e $\mathbf{v}$, $\mathbf{u}$,$\mathbf{v} \in \Sigma^*$, concatenadas sem marcação (sem um símbolo separador) obedece a relação apresentada em Eq. D.2.

$$K(\mathbf{u}, \mathbf{v}) \leq K(\mathbf{u}) + K(\mathbf{v}) + 2\log\left(\min\left(K(\mathbf{u}), K(\mathbf{v})\right)\right) \tag{D.2}$$

Definição D.3. *(Complexidade Condicional de Kolmogorov) Quando já se sabe o comprimento* $l(\mathbf{u})$ *de* $\mathbf{u} \in \Sigma^*$*, então a complexidade condicional de Kolmogorov é definida como*

$$K(\mathbf{u}|l(\mathbf{u})) = \min_{p:\mathcal{U}(p,l(p))=\mathbf{u}} l(p)$$

Uma das propriedades da complexidade de Kolmogorov é que a complexidade condicional de Kolmogorov de uma palavra $\mathbf{u} \in \Sigma^*$ é menor ou igual ao comprimento da palavra. Isso é anunciado no Teorema D.1.

Teorema D.1. *(Teorema 2.1.2 de [57, p.100]) Para uma constante c, vale*

$$K(\mathbf{u}) \leq l(\mathbf{u}) + c \tag{D.3}$$

Essa desigualdade pode ser estendida para levar em consideração o tamanho do alfabeto, como pode ser observado nos corolários D.1 e D.2.

Corolário D.1.

$$K(\mathbf{u}) \leq l(\mathbf{u}) \log|\Sigma| + c \tag{D.4}$$

Corolário D.2. *Seja qualquer palavra* $\mathbf{u} \in \Sigma^*$ *e alguma constante c, vale*

$$K(\mathbf{u}) \leq l(\mathbf{u}) \log|\Sigma| + 2\log|\Sigma| + c \tag{D.5}$$

Para entender alguns dos resultados relacionados à entropia de intercâmbio apresentados por [25] no próximo capítulo, será necessário utilizar o teorema a seguir que relaciona a complexidade de Kolmogorov e a entropia.

Teorema D.2. *(Relação entre Complexidade de Kolmogorov e Entropia [43, p.473]) Considerando o processo estocástico* $\{X_i\}$ *sendo iid de acordo com a função de massa de probabilidade* $f(x)$*,* $x \in \Sigma$*, em que* Σ *é o alfabeto finito. Denotando* $f(\mathbf{x}) = \prod_{i=1}^{n} f(x_i)$*,*

então, para todo n, existe uma constante c tal que vale

$$H(X) \leq \frac{1}{n} \sum_{\mathbf{x}} f(\mathbf{x}) K(\mathbf{x}|n) \leq H(X) + \frac{(|\Sigma| - 1) \log n}{n} + \frac{c}{n}. \tag{D.6}$$

Por conseguinte,

$$\mathbb{E}\left[\frac{1}{n} K(\mathbf{X}|n)\right] \to H(X). \tag{D.7}$$

em que $\mathbb{E}[\cdot]$ é o valor esperado.

Ao remover a dependência do tamanho n da sequência, considerando $n \to \infty$, pode-se interpretar que, conforme os comprimentos da entrada aumentam, o computador consegue comprimir as entradas de forma que a taxa se aproxima cada vez mais da entropia $H(X)$, e

$$\mathbb{E}\left[\frac{1}{n} K(\mathbf{X})\right] \to H(X). \tag{D.8}$$

O problema com a complexidade de Kolmogorov é que ela é uma função não-computável. Isso quer dizer que não há programa e nem computador universal que retornará o valor da complexidade de Kolmogorov para determinada entrada.